MITSUBISHI

三菱シーケンサ

MELSEC Q series　MELSEC L series　MELSEC-F

MELSEC-Q/L/F
構造化プログラミングマニュアル

基礎編

●安全上のご注意●

（ご使用前に必ずお読みください）

MELSEC-Q/L/F シリーズシーケンサのご使用に際しては，各製品に付属しているマニュアルおよび付属マニュアルで紹介している関連マニュアルをよくお読みいただくと共に，安全に対して十分に注意を払って，正しい取扱いをしていただくようお願いいたします。

製品に付属しているマニュアルは必要なときに読めるよう大切に保管すると共に，必ず最終ユーザまでお届けいただくようお願いいたします。

●製品の適用について●

(1) 当社シーケンサをご使用いただくにあたりましては，万一シーケンサに故障・不具合などが発生した場合でも重大な事故にいたらない用途であること，および故障・不具合発生時にはバックアップやフェールセーフ機能が機器外部でシステム的に実施されていることをご使用の条件とさせていただきます。

(2) 当社シーケンサは，一般工業などへの用途を対象とした汎用品として設計・製作されています。したがいまして，以下のような機器・システムなどの特殊用途へのご使用については，当社シーケンサの適用を除外させていただきます。万一使用された場合は当社として当社シーケンサの品質，性能，安全に関る一切の責任（債務不履行責任，瑕疵担保責任，品質保証責任，不法行為責任，製造物責任を含むがそれらに限定されない）を負わないものとさせていただきます。
・各電力会社殿の原子力発電所およびその他発電所向けなどの公共への影響が大きい用途
・鉄道各社殿および官公庁殿など，特別な品質保証体制の構築を当社にご要求になる用途
・航空宇宙，医療，鉄道，燃焼・燃料装置，乗用移動体，有人搬送装置，娯楽機械，安全機械など生命，身体，財産に大きな影響が予測される用途
ただし，上記の用途であっても，具体的に使途を限定すること，特別な品質（一般仕様を超えた品質等）をご要求されないこと等を条件に，当社の判断にて当社シーケンサの適用可とする場合もございますので，詳細につきましては当社窓口へご相談ください。

改 訂 履 歴

※ 取扱説明書番号は，本説明書の裏表紙の左下に記載してあります。

印刷日付	※ 取扱説明書番号	改 訂 内 容
2008 年 4 月	SH(名)-080735-A	初版印刷
2008 年 7 月	SH(名)-080735-B	一部修正 本マニュアルで使用する総称・略称
2008 年 11 月	SH(名)-080735-C	機種追加 Q00UJCPU，Q00UCPU，Q01UCPU，Q10UDHCPU，Q10UDEHCPU，Q20UDHCPU，Q20UDEHCPU，FX シリーズ 一部追加 マニュアルについて 一部修正 本マニュアルで使用する総称・略称，1.3 節，4.3.3 項，4.3.4 項，4.4.1 項，4.4.2 項，付 .1
2009 年 2 月	SH(名)-080735-D	一部修正 本マニュアルで使用する総称・略称
2009 年 4 月	SH(名)-080735-E	機種追加 Q00JCPU，Q00CPU，Q01CPU 一部追加 3.5 節，3.6 節，4.4 節，4.4.1 項，4.4.2 項，4.4.3 項，4.4.4 項，4.4.5 項，4.4.6 項，4.6 節，4.8 節，付 .3 一部修正 本マニュアルの位置付け→ 1.2 節へ変更， 本マニュアルで使用する総称・略称→ 1.3 節へ変更，1.5 節，4.2.8 項，4.3.4 項，4.5 節→ 4.4.7 項へ変更，4.5.2 項，4.5.3 項，4.6 節→ 4.4.8 項へ変更，付 2
2009 年 6 月	SH(名)-080735-F	一部修正 1.3 節，5.2.2 項，付 2
2009 年 12 月	SH(名)-080735-G	機種追加 L02CPU，L26CPU-BT 一部追加 製品の適用について 一部修正 マニュアルについて，1.2 節，1.3 節，1.5 節，4.4.1 項，4.4.4 項，4.5.1 項，4.5.2 項，4.6 節
2010 年 1 月	SH(名)-080735-H	機種追加 Q50UDEHCPU，Q100UDEHCPU 一部修正 1.2 節，1.3 節，1.5 節，付 .2
2010 年 6 月	SH(名)-080735-I	一部修正 4.4.7 項，付 .2
2010 年 9 月	SH(名)-080735-J	一部修正 4.4.7 項，5.1.1 項，5.1.3 項，付 .2

印刷日付	*取扱説明書番号	改　訂　内　容
2010 年 11 月	SH(名)-080735-K	一部修正 1.2 節，5.1.1 項
2011 年 1 月	SH(名)-080735-L	一部追加 4.2.5 項，5.2.3 項，5.2.4 項 一部修正 1.2 節，4.2.2 項，4.2.3 項，4.2.4 項，4.2.8 項，4.3.4 項，4.4.4 項，4.4.7 項，4.6 節，5.1.3 項，5.1.5 項
2011 年 5 月	SH(名)-080735-M	機種追加 L02CPU-P，L26CPU-PBT 一部追加 5.2.5 項 一部修正 1.3 節，1.4 節，1.5 節，4.2.6 項，4.2.7 項，4.4.2 項，4.4.3 項，4.4.4 項，5.1.3 項，5.1.6 項，5.2 節，5.2.2 項
2012 年 2 月	SH(名)-080735-N	機種追加 FX3GC 一部修正 マニュアルについて，1.2 節，1.3 節，1.5 節，4.3.5 項，4.4.2 項，4.4.3 項，4.4.4 項，4.4.6 項，4.5.1 項，4.5.3 項，4.6 節，付 2
2012 年 5 月	SH(名)-080735-O	一部修正 マニュアルについて，4.1.3 項，5.2.4 項
2012 年 11 月	SH(名)-080735-P	プロセス CPU，二重化 CPU，ユニバーサルモデル QCPU，LCPU の機種追加に伴う全面見直し 機種追加 Q02PHCPU，Q06PHCPU，Q12PHCPU，Q25PHCPU，Q12PRHCPU，Q25PRHCPU，Q03UDVCPU，Q04UDVCPU，Q06UDVCPU，Q13UDVCPU，Q26UDVCPU，L02SCPU，L26CPU
2013 年 1 月	SH(名)-080735-Q	機種追加 L06CPU 一部修正 1.3 節，1.5 節
2013 年 6 月	SH(名)-080735-R	機種追加 Q04UDPVCPU，Q06UDPVCPU，Q13UDPVCPU，Q26UDPVCPU，L02SCPU-P，L06CPU-P，L26CPU-P，FX3S 一部修正 1.2 節，1.3 節，1.5 節，4.3.5 項，4.4.4 項，4.6 節

本書によって，工業所有権その他の権利の実施に対する保証，または実施権を許諾するものではありません。また
本書の掲載内容の使用により起因する工業所有権上の諸問題については，当社は一切その責任を負うことができません。

© 2008 MITSUBISHI ELECTRIC CORPORATION

はじめに

このたびは，三菱シーケンサ MELSEC-Q/L/F シリーズをお買い上げいただきまことにありがとうございました。
ご使用前に本マニュアルおよび関連マニュアルをよくお読みいただき，プログラミングの仕様を十分ご理解のうえ，正しくご使用くださるようお願い致します。
本マニュアルで紹介するプログラム例を実際のシステムへ流用する場合は，対象システムにおける制御に問題がないことを十分検証ください。

目　次

安全上のご注意 ····· A - 1
製品の適用について ····· A - 2
改　訂　履　歴 ····· A - 3
はじめに ····· A - 5
目　次 ····· A - 5
マニュアルについて ····· A - 8

1章　概　要　1 - 1 ～ 1 - 8

1.1　概要　1 - 2
1.2　本マニュアルの位置付け　1 - 2
1.3　本マニュアルで使用する総称・略称　1 - 6
1.4　構造化プログラムの特長　1 - 7
1.5　対応する CPU ユニット　1 - 8
1.6　対応するソフトウェアパッケージ　1 - 8

2章　シーケンスプログラムの構造化設計　2 - 1 ～ 2 - 4

2.1　シーケンスプログラムの階層化とは　2 - 2
2.2　シーケンスプログラムの部品化とは　2 - 3

3章　プログラム作成の手順　3 - 1 ～ 3 - 2

3.1　構造化プロジェクトのシーケンスプログラム作成手順　3 - 2

4章　プログラム構成　4 - 1 ～ 4 - 58

4.1　プログラム構成の概要　4 - 2
4.1.1　プロジェクト ····· 4 - 3
4.1.2　プログラムファイル ····· 4 - 3
4.1.3　タスク ····· 4 - 4
4.2　プログラム部品　4 - 5
4.2.1　プログラム部品の種類 ····· 4 - 5
4.2.2　プログラム ····· 4 - 6
4.2.3　ファンクション ····· 4 - 6
4.2.4　ファンクションブロック ····· 4 - 7
4.2.5　オペレータ ····· 4 - 8

4.2.6 回路ブロック ····· 4 - 9
4.2.7 プログラム部品のプログラム言語 ····· 4 - 10
4.2.8 ファンクション，ファンクションブロック，オペレータの違い ····· 4 - 11
4.2.9 EN/ENO の考え方 ····· 4 - 14
4.3 ラベル 4 - 16
4.3.1 グローバルラベル ····· 4 - 16
4.3.2 ローカルラベル ····· 4 - 16
4.3.3 ラベルのクラス ····· 4 - 17
4.3.4 ラベルの設定 ····· 4 - 18
4.3.5 データ型 ····· 4 - 19
4.3.6 定数の表記方法 ····· 4 - 21
4.4 データの指定方法 4 - 22
4.4.1 ビットデータの場合 ····· 4 - 23
4.4.2 ワード（16 ビット）データの場合 ····· 4 - 24
4.4.3 ダブルワード（32 ビット）データの場合 ····· 4 - 26
4.4.4 単精度実数／倍精度実数データの場合 ····· 4 - 29
4.4.5 文字列データの場合 ····· 4 - 33
4.4.6 時間データの場合 ····· 4 - 34
4.4.7 配列 ····· 4 - 35
4.4.8 構造体 ····· 4 - 38
4.5 デバイスとアドレス 4 - 39
4.5.1 デバイス ····· 4 - 39
4.5.2 アドレス ····· 4 - 40
4.5.3 デバイス表記とアドレス表記 ····· 4 - 41
4.6 インデックス修飾 4 - 44
4.7 ライブラリ 4 - 56
4.7.1 ユーザライブラリ ····· 4 - 57
4.8 名称を付けるときの注意事項 4 - 58

5 章 プログラムの記述方法 5 - 1 ～ 5 - 20

5.1 ST 5 - 2
5.1.1 基本書式 ····· 5 - 2
5.1.2 ST の演算子 ····· 5 - 3
5.1.3 ST の構文 ····· 5 - 4
5.1.4 ST でのファンクションの呼び出し ····· 5 - 9
5.1.5 ST でのファンクションブロックの呼び出し ····· 5 - 10
5.1.6 条件文・反復文使用時の注意事項 ····· 5 - 11
5.2 構造化ラダー /FBD 5 - 13
5.2.1 基本書式 ····· 5 - 13
5.2.2 構造化ラダー /FBD の回路記号 ····· 5 - 14
5.2.3 実行順序 ····· 5 - 16
5.2.4 回路分岐におけるコンパイル結果 ····· 5 - 17
5.2.5 構造化ラダー /FBD プログラム作成時の注意事項 ····· 5 - 19

付 録 付 - 1 ～ 付 - 14

付 .1 普遍的データ型に対応するデバイス 付 - 2
付 .2 ラベル名やデータ名で使用できない文字列 付 - 6

付 .3　ラダーからの置換え　付 - 9
付 .3.1　作成手順 ・・・

マニュアルについて

関連マニュアル

本製品に関連するマニュアルには，下記のものがあります。

各マニュアルは，必要に応じて本表を参考にご依頼ください。

(1) 構造化プログラミング

マニュアル名称	マニュアル番号（形名コード）	標準価格
MELSEC-Q/L 構造化プログラミングマニュアル（共通命令編） 構造化プログラムで使用できる，シーケンス命令，基本命令，応用命令などの共通命令に関する仕様，機能について説明しています。（別売）	SH-080736 (13JC18)	¥4,000
MELSEC-Q/L 構造化プログラミングマニュアル（応用関数編） 構造化プログラムで使用できる，応用関数に関する仕様，機能について説明しています。（別売）	SH-080737 (13JC19)	¥3,000
MELSEC-Q/L 構造化プログラミングマニュアル（特殊命令編） 構造化プログラムで使用できる，ユニット専用命令，PID 制御命令，内蔵 I/O 機能用命令などの特殊命令に関する仕様，機能について説明しています。（別売）	SH-080738 (13JC20)	¥3,000
FXCPU 構造化プログラミングマニュアル [デバイス・共通説明編] GX Works2 で提供される構造化プログラム用デバイス，パラメータを説明しています。（別売）	JY997D30801 (09R920)	¥2,400
FXCPU 構造化プログラミングマニュアル [シーケンス命令編] GX Works2 で提供される構造化プログラム用シーケンス命令を説明しています。（別売）	JY997D29601 (09R921)	¥3,000
FXCPU 構造化プログラミングマニュアル [応用関数編] GX Works2 で提供される構造化プログラム用応用関数を説明しています。（別売）	JY997D25101 (09R922)	¥2,400

(2) GX Works2 の操作

マニュアル名称	マニュアル番号（形名コード）	標準価格
GX Works2 Version 1 オペレーティングマニュアル （共通編） GX Works2 のシステム構成，パラメータ設定，オンライン機能の操作方法など，シンプルプロジェクトと構造化プロジェクトに共通な機能を記載しています。（別売）	SH-080730 (13JV90)	¥4,000
GX Works2 Version 1 オペレーティングマニュアル （構造化プロジェクト編） GX Works2 の構造化プロジェクトでのプログラム作成，モニタなどの操作方法について記載しています。（別売）	SH-080732 (13JV92)	¥3,000
はじめよう GX Works2 （構造化プロジェクト編） はじめて GX Works2 を使用する方のために，構造化プロジェクトでのプログラム作成から編集，モニタなどの基本的な操作方法について説明しています。（別売）	SH-080734 (13JY65)	¥1,500

ポイント

オペレーティングマニュアルは，ソフトウェアパッケージの CD-ROM に PDF ファイルで格納されています。単品でマニュアルを希望される場合は，印刷物を別売で用意していますので上記表のマニュアル番号（形名コード）にてご用命願います。

1 概　要

1.1　概要 1-2
1.2　本マニュアルの位置付け 1-2
1.3　本マニュアルで使用する総称・略称 1-6
1.4　構造化プログラムの特長 1-7
1.5　対応する CPU ユニット 1-8
1.6　対応するソフトウェアパッケージ 1-8

1.1 概要

本マニュアルでは，構造化プログラミング手法を使ってシーケンスプログラムを作成するために必要なプログラムの構成と内容，プログラムの記述方法などの基礎知識について，記載しています。

1.2 本マニュアルの位置付け

本マニュアルでは，構造化プログラムの作成に必要なプログラミング方法，プログラム言語の種類などについて説明しています。

目的に応じた参照先マニュアルを下記に示します。

各マニュアルの記載内容，マニュアル番号などについては「関連マニュアル」一覧を参照してください。

(1) GX Works2 の操作

目的		GX Works2 インストール手順書	はじめよう GX Works2		GX Works2 Version 1 オペレーティングマニュアル			
			シンプルプロジェクト編	構造化プロジェクト編	共通編	シンプルプロジェクト編	構造化プロジェクト編	インテリジェント機能ユニット操作編
インストール	動作環境，インストール方法を知りたい							
	USB ドライバのインストール方法を知りたい							
GX Works2 の各種操作	GX Works2 のすべての機能が知りたい							
	GX Works2 のプロジェクト種別と使用できる言語が知りたい							
	はじめてシンプルプロジェクトを使用するときの基本の操作と手順を知りたい							
	はじめて構造化プロジェクトを使用するときの基本の操作と手順を知りたい							
	プロジェクト種別に関係なく使用できる機能の操作方法を知りたい							
	プログラミング用の機能と操作方法を知りたい							
	インテリジェント機能ユニットのデータ設定方法を知りたい							

(2) プログラミングで使用する言語ごとの操作

各言語のプログラミングで使用する命令の詳細は (3) を参照してください。

目的		はじめよう GX Works2		GX Works2 Version 1 オペレーティングマニュアル	
		シンプル プロジェクト編	構造化 プロジェクト編	シンプル プロジェクト編	構造化 プロジェクト編
シンプル プロジェクト	ラダー	概要		詳細	
	SFC	*1 概要		詳細	
	ST		概要		詳細
構造化 プロジェクト	ラダー	概要		詳細	
	SFC	*1 概要		詳細	
	構造化ラダー /FBD		概要		詳細
	ST		概要		詳細

*1: MELSAP3，FX シリーズ用 SFC のみ

(3) 各言語のプログラミングで使用する命令の詳細（QCPU(Q モード)/LCPU の場合 ）

目的		MELSEC-Q/L/F 構造化プログラミングマニュアル	MELSEC-Q/L 構造化プログラミングマニュアル			MELSEC-Q/L プログラミングマニュアル	MELSEC-Q/L/QnA プログラミングマニュアル		MELSEC-Q プログラミング／構造化プログラミングマニュアル	使用するユニットのマニュアル
		基礎編	共通命令編	特殊命令編	応用関数編	共通命令編	PID 制御命令編	SFC 編	プロセス制御命令編	-
すべての言語	シーケンサ CPU のエラーコード，特殊リレー・特殊レジスタの内容を知りたい									*1 詳細
ラダーを使用する場合	共通命令の種類と詳細を知りたい					詳細				
	インテリジェント機能ユニット用命令の種類と詳細を知りたい									詳細
	ネットワークユニット用命令の種類と詳細を知りたい									詳細
	PID 制御機能用命令の種類と詳細を知りたい						詳細			
	プロセス制御命令の種類と詳細を知りたい								詳細	
SFC を使用する場合	SFC(MELSAP3) の仕様，機能，命令などの詳細を知りたい							詳細		
構造化ラダー / FBD または ST を使用する場合	構造化プログラミングするための基礎を知りたい	詳細								
	共通命令の種類と詳細を知りたい		詳細							
	インテリジェント機能ユニット用命令の種類と詳細を知りたい			概要						詳細
	ネットワークユニット用命令の種類と詳細を知りたい			概要						詳細
	PID 制御機能用命令の種類と詳細を知りたい			概要			詳細			
	応用関数の種類と詳細を知りたい				詳細					
	プロセス制御命令の種類と詳細を知りたい								詳細	

*1： 使用している CPU ユニットのユーザーズマニュアル（ハードウェア設計・保守点検編）を参照してください。

（4）各言語のプログラミングで使用する命令の詳細（FXCPU の場合 ）

目的		MELSEC-Q/L/F 構造化プログラミングマニュアル	FXCPU 構造化プログラミングマニュアル			FXCPU プログラミングマニュアル		
		基礎編	デバイス・共通説明編	シーケンス命令編	応用関数編	FX0,FX0S,FX0N,FX1,FX2,FX2C	FX1S,FX1N,FX2N,FX1NC,FX2NC	FX3S,FX3G,FX3U,FX3GC,FX3UC
ラダーを使用する場合	基本・応用命令の種類と詳細，デバイスとパラメータの内容を知りたい					詳細	詳細	詳細
SFC を使用する場合	SFC の仕様，機能，命令などの詳細を知りたい					詳細	詳細	詳細
構造化ラダー / FBD または ST を使用する場合	構造化プログラミングするための基礎を知りたい	詳細						
	デバイスとパラメータ，エラーコードの内容を知りたい		詳細					
	シーケンス命令の種類と詳細を知りたい			詳細				
	応用関数の種類と詳細を知りたい				詳細			

1.3 本マニュアルで使用する総称・略称

本マニュアルでは，ソフトウェアパッケージ，シーケンサ CPU などを，次の総称・略称で表しています。対象形名の明示が必要なときは，ユニット形名を記載しています。

総称／略称	総称・略称の内容
GX Works2	MELSEC シーケンサソフトウェアパッケージの製品名
GX Developer	
ベーシックモデル QCPU	Q00JCPU，Q00CPU，Q01CPU の総称。
ハイパフォーマンスモデル QCPU	Q02CPU，Q02HCPU，Q06HCPU，Q12HCPU，Q25HCPU の総称。
プロセス CPU	Q02PHCPU，Q06PHCPU，Q12PHCPU，Q25PHCPU の総称。
二重化 CPU	Q12PRHCPU，Q25PRHCPU の総称。
ユニバーサルモデル QCPU	Q00UJCPU，Q00UCPU，Q01UCPU，Q02UCPU，Q03UDCPU，Q03UDVCPU，Q03UDECPU，Q04UDHCPU，Q04UDVCPU，Q04UDPVCPU，Q04UDEHCPU，Q06UDHCPU，Q06UDVCPU，Q06UDPVCPU，Q06UDEHCPU，Q10UDHCPU，Q10UDEHCPU，Q13UDHCPU，Q13UDVCPU，Q13UDPVCPU，Q13UDEHCPU，Q20UDHCPU，Q20UDEHCPU，Q26UDHCPU，Q26UDVCPU，Q26UDPVCPU，Q26UDEHCPU，Q50UDEHCPU，Q100UDEHCPU の総称。
ユニバーサルモデル 高速タイプ QCPU	Q03UDVCPU，Q04UDVCPU，Q06UDVCPU，Q13UDVCPU，Q26UDVCPU の総称。
ユニバーサルモデル プロセス CPU	Q04UDPVCPU，Q06UDPVCPU，Q13UDPVCPU，Q26UDPVCPU の総称。
QCPU(Q モード)	ベーシックモデル QCPU，ハイパフォーマンスモデル QCPU，プロセス CPU，二重化 CPU，ユニバーサルモデル QCPU の総称。
LCPU	L02SCPU，L02SCPU-P，L02CPU，L02CPU-P，L06CPU，L06CPU-P，L26CPU，L26CPU-P，L26CPU-BT，L26CPU-PBT の総称。
FXCPU	MELSEC-FX シリーズシーケンサの総称。 (対象は，FX_{0S}，FX_{0}，FX_{0N}，FX_{1}，FX_{1S}，FX_{1N}，FX_{1NC}，FX_{2}，FX_{2C}，FX_{2N}，FX_{2NC}，FX_{3S}，FX_{3G}，FX_{3GC}，FX_{3U}，FX_{3UC} です)
CPU ユニット	QCPU(Q モード)，LCPU，FXCPU の総称。
QnU(D)(H)CPU	Q02UCPU，Q03UDCPU，Q04UDHCPU，Q06UDHCPU，Q10UDHCPU，Q13UDHCPU，Q20UDHCPU，Q26UDHCPU の総称。
QnUDVCPU	Q03UDVCPU，Q04UDVCPU，Q06UDVCPU，Q13UDVCPU，Q26UDVCPU の総称。
QnUDPVCPU	Q04UDPVCPU，Q06UDPVCPU，Q13UDPVCPU，Q26UDPVCPU の総称。
QnUDE(H)CPU	Q03UDECPU，Q04UDEHCPU，Q06UDEHCPU，Q10UDEHCPU，Q13UDEHCPU，Q20UDEHCPU，Q26UDEHCPU，Q50UDEHCPU，Q100UDEHCPU の総称。
パソコン	Windows® が動作するパーソナルコンピュータの総称。
IEC61131-3	国際標準規格 IEC 61131-3 規格の略称。
共通命令	シーケンス命令，基本命令，応用命令，データリンク用命令，マルチ CPU 間専用命令，マルチ CPU 間高速通信専用命令，二重化システム用命令の総称。
特殊命令	ユニット専用命令，PID 制御命令，ソケット通信機能用命令，内蔵 I/O 機能用命令，データロギング機能用命令の総称。
応用関数	IEC61131-3 で定義された関数（ファンクションおよびファンクションブロック）の総称。 (シーケンサの内部では，複数の共通命令を組み合わせて実行される。)

1.4 構造化プログラムの特長

構造化プログラムの特長を下記に示します。

(1) 構造化設計

構造化設計とは，シーケンサ CPU で行う制御内容を階層的な構造にできるように小さな処理単位（部品）に分けて，プログラミングする手法です。構造化プログラムでは，シーケンスプログラムの構造化を意識して，設計することができます。

プログラムを階層化するメリットは，下記のとおりです。

- 初めはプログラムの概要を検討し，徐々に詳細な設計をしていくことができます。
- 階層化で設計された最下位のプログラムは，きわめて単純で独立性の高いものになります。

プログラムを部品化するメリットは，下記のとおりです。

- 部品ごとの処理が明確になるため全体を見通し良くできます。
- プログラムを分担し，複数人で作成することができます。
- プログラムの再利用性が高まり，開発効率が上がります。

(2) 複数のプログラム言語

構造化プログラムでは，複数のプログラム言語が用意されています。ユーザは用途に応じて，最適なプログラム言語を選択し，組み合わせて使用できます。
また，プログラム部品ごとに異なった言語でプログラミングできます。

表 1.4-1 構造化プログラムで使用できるプログラム言語

名称		説明
ST（ストラクチャードテキスト）		C 言語などに似た計算機技術者向きのテキスト言語
構造化ラダー		接点やコイルなどで回路を表すグラフィック言語
	FBD	ファンクションやファンクションブロックを罫線で接続して回路を表すグラフィック言語

プログラム言語の概要は下記を参照してください。
☞ 4.2.7 項　プログラム部品のプログラム言語
プログラム言語ごとの記述は下記を参照してください。
☞ 第 5 章　プログラムの記述方法
従来の GX Developer や，GX Works2 のシンプルプロジェクトで使用しているラダー／SFC 言語を使用することができます。
記述方法は，下記を参照してください。
☞ 各 CPU に対応するプログラミングマニュアル

(3) プログラムの再利用性の向上

部品化したプログラムは，ライブラリとして保存できます。プログラムをライブラリ化することでプログラム資産の共有化を図り，プログラムの再利用性向上に役立ちます。

1.5 対応する CPU ユニット

構造化プロジェクトのプログラムは，下記 CPU ユニットに対応しています。

表 1.5-1 対応する CPU ユニット

PC タイプ	
ベーシックモデル QCPU	Q00JCPU，Q00CPU，Q01CPU
ハイパフォーマンスモデル QCPU	Q02CPU，Q02HCPU，Q06HCPU，Q12HCPU，Q25HCPU
プロセス CPU	Q02PHCPU，Q06PHCPU，Q12PHCPU，Q25PHCPU
二重化 CPU	Q12PRHCPU，Q25PRHCPU
ユニバーサルモデル QCPU	Q00UJCPU，Q00UCPU，Q01UCPU，Q02UCPU，Q03UDCPU，Q03UDVCPU，Q03UDECPU，Q04UDHCPU，Q04UDVCPU，Q04UDPVCPU，Q04UDEHCPU，Q06UDHCPU，Q06UDVCPU，Q06UDPVCPU，Q06UDEHCPU，Q10UDHCPU，Q10UDEHCPU，Q13UDHCPU，Q13UDVCPU，Q13UDPVCPU，Q13UDEHCPU，Q20UDHCPU，Q20UDEHCPU，Q26UDHCPU，Q26UDVCPU，Q26UDPVCPU，Q26UDEHCPU，Q50UDEHCPU，Q100UDEHCPU
LCPU	L02SCPU，L02SCPU-P，L02CPU，L02CPU-P，L06CPU，L06CPU-P，L26CPU，L26CPU-P，L26CPU-BT，L26CPU-PBT
FXCPU	FX_{0S}，FX_{0}，FX_{0N}，FX_{1S}，FX_{1N}，FX_{1NC}，FX_{2}，FX_{2C}，FX_{2N}，FX_{2NC}，FX_{3S}，FX_{3G}，FX_{3GC}，FX_{3U}，FX_{3UC}

1.6 対応するソフトウェアパッケージ

構造化プロジェクトのプログラムの作成・編集・モニタは，下記プログラミングツールで行います。

表 1.6-1 対応するソフトウェアパッケージ

ソフトウェアパッケージ名	形　名
GX Works2	SW1DNC-GXW2-J

(1) GX Works2 とは

GX Works2 は，シーケンスプログラムの編集，デバッグ，シーケンサ CPU のモニタなどを行うソフトウェアパッケージです。パソコンの Microsoft® Windows® Operating System 環境で動作します。

ユーザが作成するシーケンスプログラムは，使用するシーケンサ CPU ごとに，「プロジェクト」という単位で管理されます。プロジェクトは，「シンプルプロジェクト」と「構造化プロジェクト」に大別されます。

ポイント

本マニュアルでは，GX Works2 の構造化プロジェクトをもとに，プログラミングの基本について説明します。

2 シーケンスプログラムの構造化設計

2.1 シーケンスプログラムの階層化とは 2-2
2.2 シーケンスプログラムの部品化とは 2-3

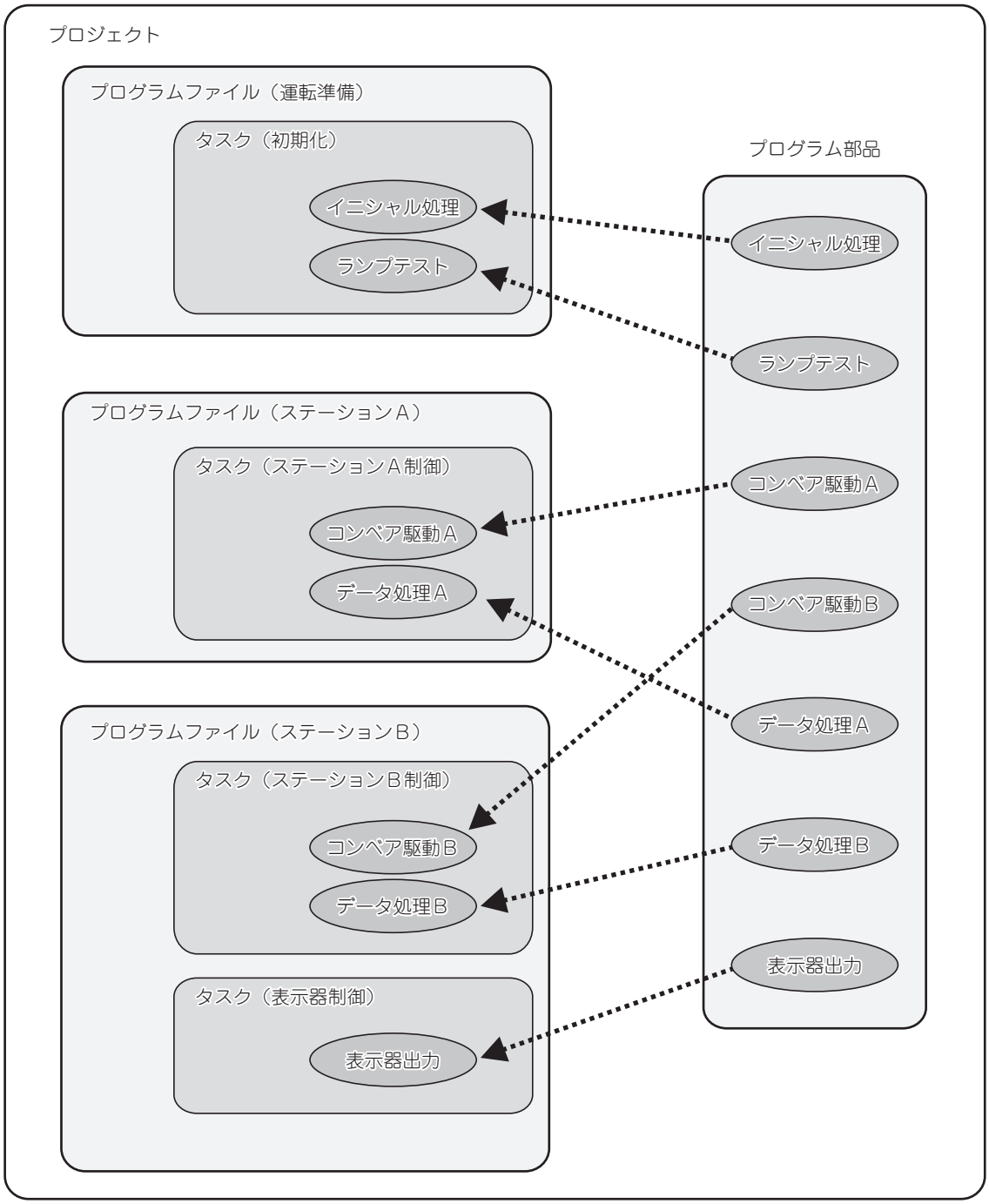
2.1 シーケンスプログラムの階層化とは

階層化とは，シーケンサ CPU で行う制御をいくつかの階層に分けて，シーケンスプログラムを作成することです。

上位の階層では，決められた範囲の処理の順番やタイミングを制御します。

階層が上位から下位へ移るごとに，決められた範囲での制御内容および処理を細分化していき，下位の階層では具体的な処理を記述します。

構造化プロジェクトでは，最上位の階層をプロジェクトとして，プログラムファイル，タスク，プログラム部品という構成で，シーケンスプログラムを階層化します。



2.2 シーケンスプログラムの部品化とは

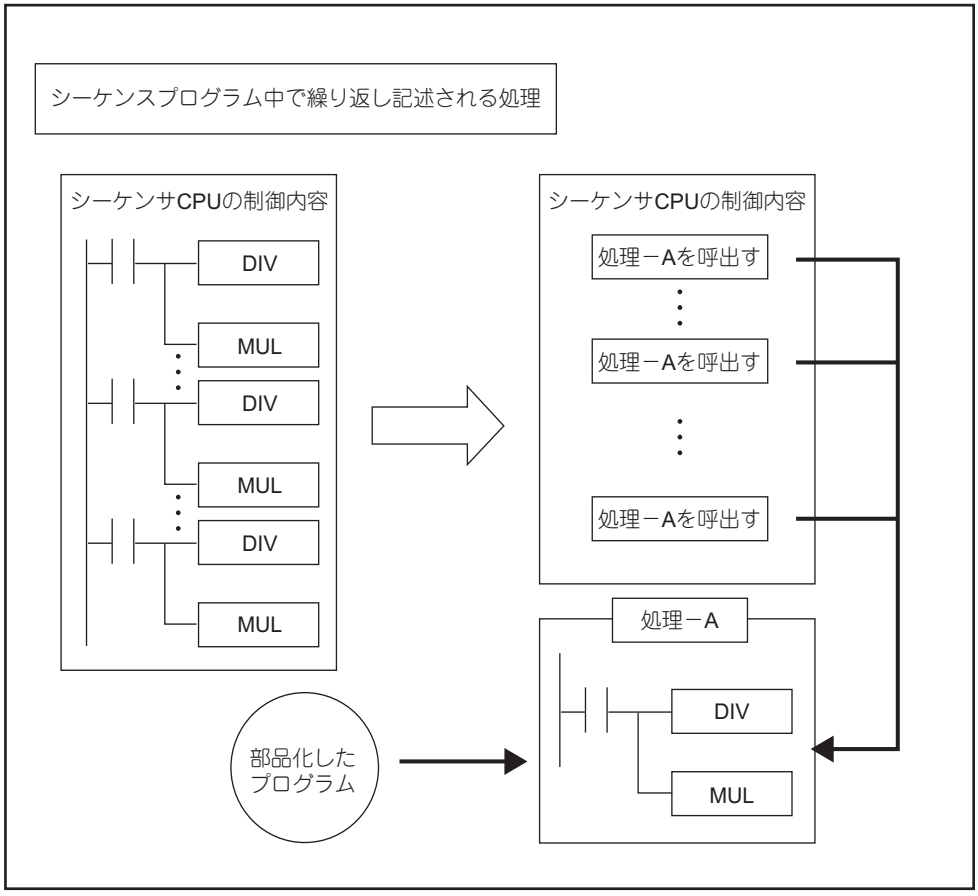
部品化とは，シーケンスプログラムを階層化したときの下位の処理を，処理内容や機能ごとにいくつかの単位に分け，単位ごとのプログラムを作成することです。

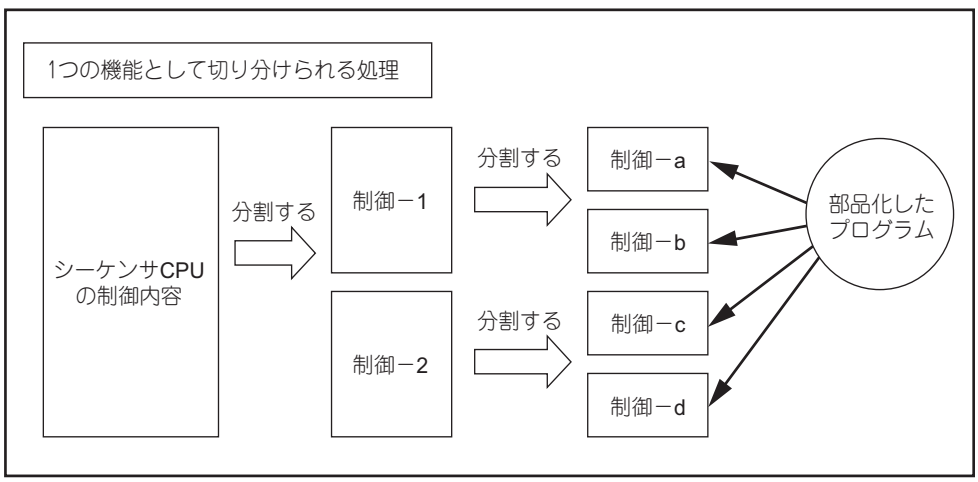
構造化設計では，下位の階層の処理を可能な限り部品化することを推奨しています。

部品は，独立性が高く，追加や交換が容易にできるように設計します。

部品化するとよい処理には下記のようなものがあります。

・ シーケンスプログラム中で繰り返し記述される処理
・ 1 つの機能として切り分けられる処理





メ　モ

3 プログラム作成の手順

3.1　構造化プロジェクトのシーケンスプログラム作成手順. 3-2

3.1 構造化プロジェクトのシーケンスプログラム作成手順

構造化プロジェクトのシーケンスプログラムを作成するときの，基本的な手順を示します。

1. プログラム構成の作成

手順
プログラムファイルを作成します。
タスクを作成します。

⇩

2. プログラム部品の作成

手順
プログラム部品を作成します。
グローバルラベルを定義します。
ローカルラベルを定義します。
各プログラム部品のプログラムを編集します。

⇩

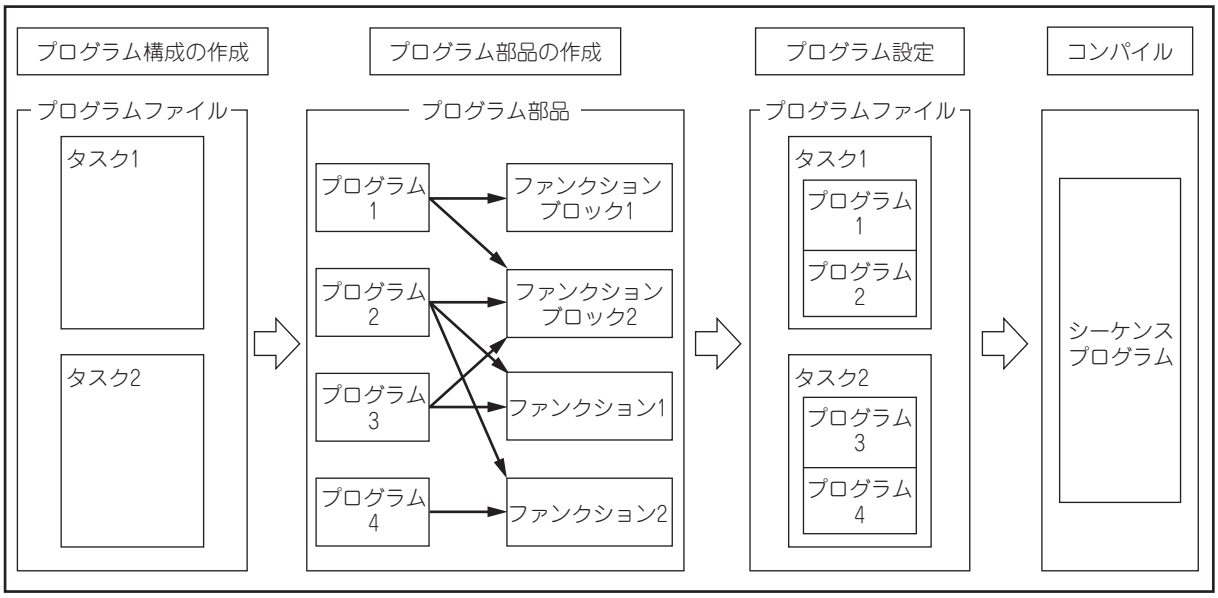
3. プログラム設定

手順
タスクにプログラム部品を登録します。

⇩

4. コンパイル

手順
コンパイルします。



4 プログラム構成

4.1 プログラム構成の概要 4-2
4.2 プログラム部品 4-5
4.3 ラベル 4-16
4.4 データの指定方法 4-22
4.5 デバイスとアドレス 4-39
4.6 インデックス修飾 4-44
4.7 ライブラリ 4-56
4.8 名称を付けるときの注意事項 4-58

4.1 プログラム構成の概要

構造化プロジェクトで作成するシーケンスプログラムは，プログラムファイル，タスク，プログラム部品によって構成されています。

それぞれの詳細は，下記を参照してください。

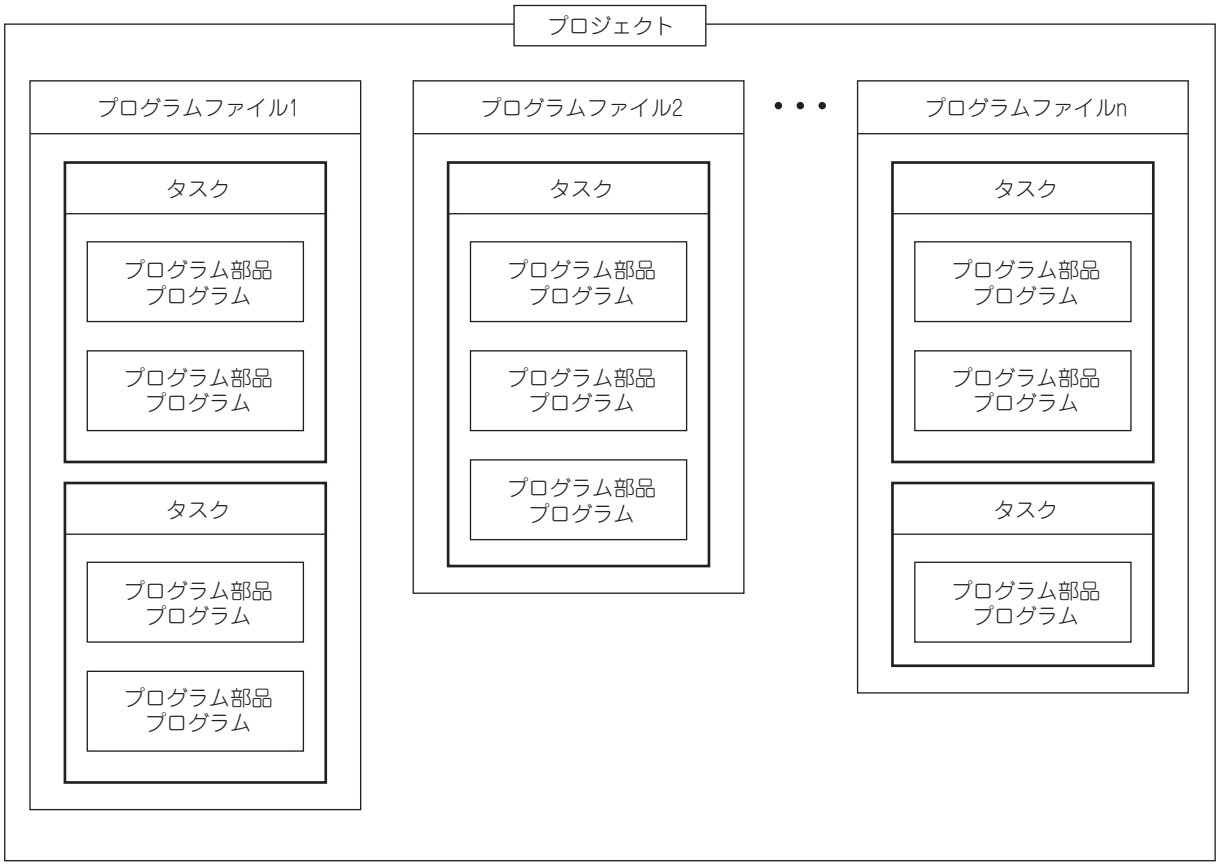
プロジェクトは　☞ 4.1.1 項　プロジェクト

プログラムファイルは　☞ 4.1.2 項　プログラムファイル

タスクは　☞ 4.1.3 項　タスク

プログラム部品は　☞ 4.2 節　プログラム部品

下図に，プロジェクトとプログラムファイル，タスク，プログラム部品の関係を示します。



4.1.1 プロジェクト

プロジェクトとは，シーケンサ CPU で実行するデータ（プログラム，パラメータなど）の総称です。

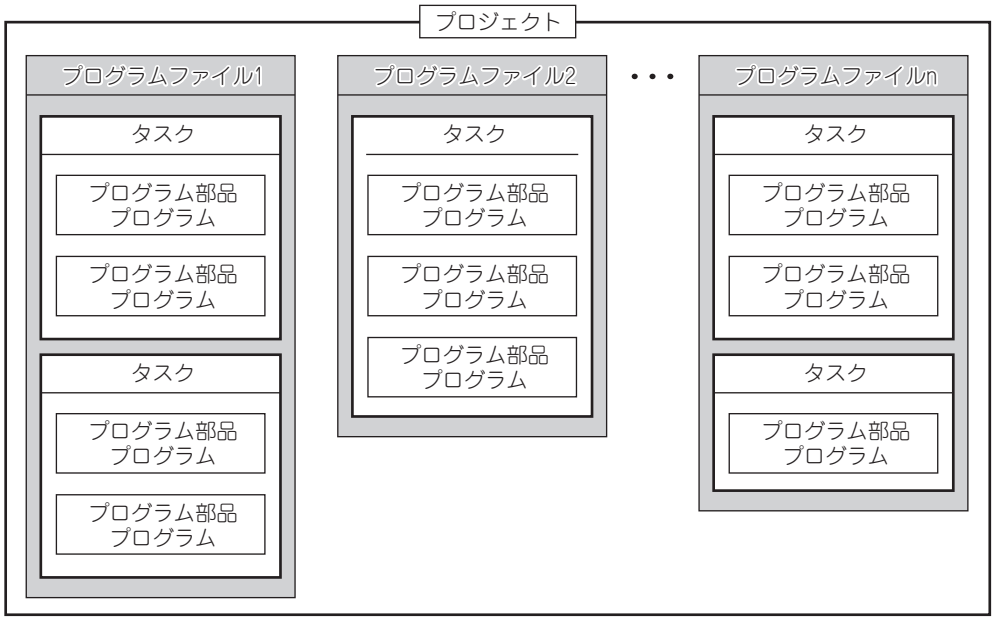
プロジェクトにはプログラムファイルを 1 つ以上作成することが必要です。

4.1.2 プログラムファイル

プログラムファイルには，1 つ以上のタスクを作成することが必要です。（作成したタスクは，プログラムファイルの制御下で実行します）

プログラムファイルをシーケンサ CPU で実行するときの実行タイプ（スキャン実行，定周期実行など）は，パラメータのプログラム設定で設定します。

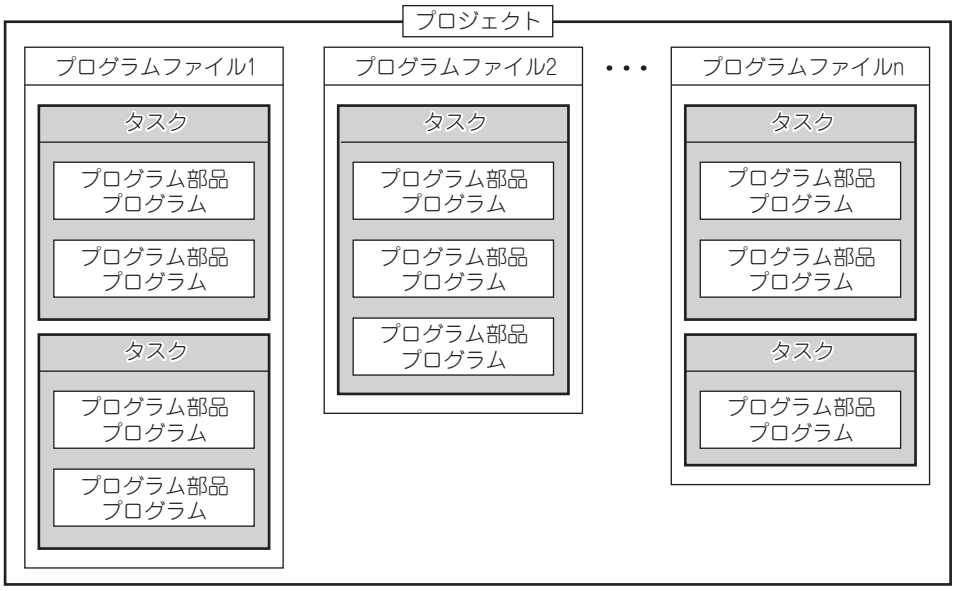
パラメータで設定する実行タイプは，使用している CPU ユニットのユーザーズマニュアルを参照してください。



4.1.3 タスク

タスクとは，複数のプログラム部品をまとめ，プログラムファイルで登録する要素です。

タスクには，プログラム部品のうちのプログラムを 1 つ以上登録することが必要です。（ファンクションとファンクションブロックは，タスクに登録できません。）



(1) タスクの実行条件

プログラムファイルに登録する各タスクには，シーケンサ CPU での実行条件を設定します。実行条件の設定により，各タスクの実行の仕方が決まります。

タスクの実行条件には下記の種類があります。

(a) 常時実行（デフォルトの実行条件）

常時実行するタスクは，スキャンごとに登録されたプログラムを実行します。

(b) イベント起動

対応するデバイスまたはラベルに値がセットされたときに実行します。

(c) 周期起動

一定の周期でタスクを実行します。

それぞれのタスクには優先度が設定できます。

● 優先度

複数のタスクで，実行条件が同時に成立した場合の，タスクの実行の優先順位を設定できます。

優先度の値がもっとも小さいタスクから順に実行されます。

優先度が同じ値の場合は，タスクのデータ名順に実行されます。

4.2 プログラム部品

プログラム部品とは，機能ごとに分けて定義されたプログラムの単位です。

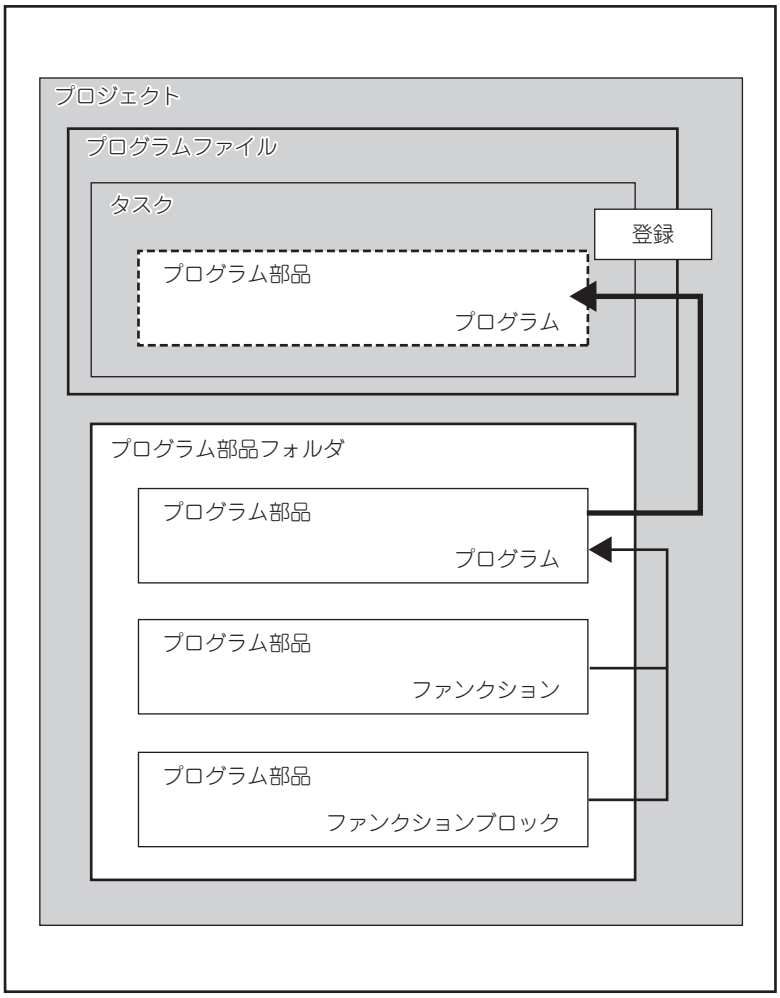
4.2.1 プログラム部品の種類

プログラム部品は，定義する内容に応じて下記の 3 種類から選択できます。

・ プログラム
・ ファンクション
・ ファンクションブロック

各プログラム部品は，プログラムとローカルラベル[*1] で構成されます。

各プログラム部品では，制御に合わせたプログラム言語で処理を記述できます。



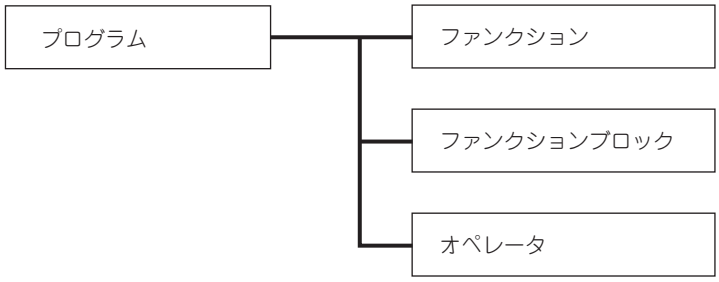
*1: ローカルラベルとは，そのプログラム部品のプログラムの中でのみ使用できるラベルです。ローカルラベルについては下記を参照してください。

☞ 4.3.2 項 ローカルラベル

4.2.2 プログラム

プログラムは，プログラム部品のうちでもっとも上位にあたる要素です。

ファンクション，ファンクションブロック，オペレータを使って編集します。



シーケンサ CPU で実行するシーケンスプログラムは，プログラム部品のプログラムで作成します。

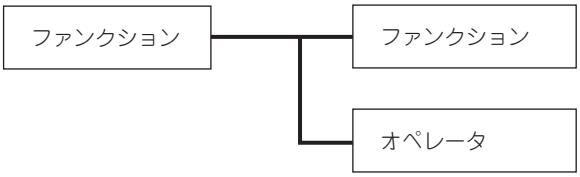
もっとも単純なシーケンスプログラムの場合，1 つのプログラムだけを作成し，1 つのタスクに登録することで，シーケンサ CPU で実行させることができます。

プログラムは，プログラム言語の ST，構造化ラダー /FBD で記述できます。

4.2.3 ファンクション

ファンクションは，ファンクション，オペレータを使って編集します。

プログラムやファンクション，ファンクションブロックから呼び出して使用することができます。



ファンクションは，同じ入力に対して，処理の結果として，常に同じ値を出力します。

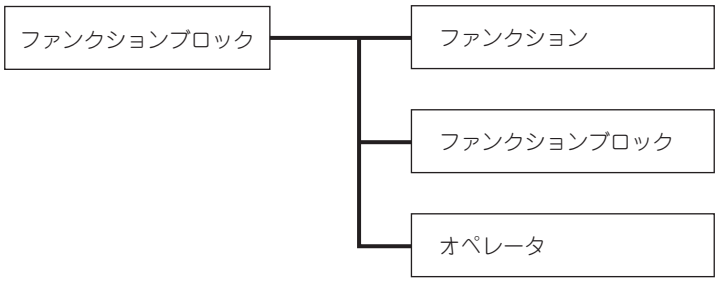
単純で独立した，良く使うアルゴリズムを定義しておくと，有効に再利用できます。

ファンクションは，プログラム言語の ST，構造化ラダー /FBD で記述できます。

4.2.4 ファンクションブロック

ファンクションブロックはファンクション，ファンクションブロック，オペレータを使って編集します。

ファンクションブロックは，プログラムやファンクションブロックから呼び出して使用することができます。ファンクションから呼び出すことはできません。



ファンクションブロックは内部変数と出力変数に値を保存できるので，入力された状態を保持することができます。保持した値を次回の処理に用いるため，同じ入力値でも毎回同じ結果を出力するとは限りません。

ファンクションブロックは，プログラム言語の ST，構造化ラダー /FBD で記述できます。

● インスタンス化
プログラムでファンクションブロックを使用するためには，ファンクションブロックをインスタンス化する必要があります。
インスタンスの詳細は ☞ 4.2.8 項 ファンクション，ファンクションブロック，オペレータの違い

ポイント

インスタンスとは，ファンクションブロックのラベルに割り付けられたデバイスを総称した変数です。
ローカルラベルでインスタンスを作成した場合，デバイスが自動割付けされます。

4.2.5 オペレータ

オペレータは，プログラム，ファンクション，ファンクションブロックから呼び出して使用することができます。編集することはできません。

同じ入力に対して，処理の結果として常に同じ値を出力します。

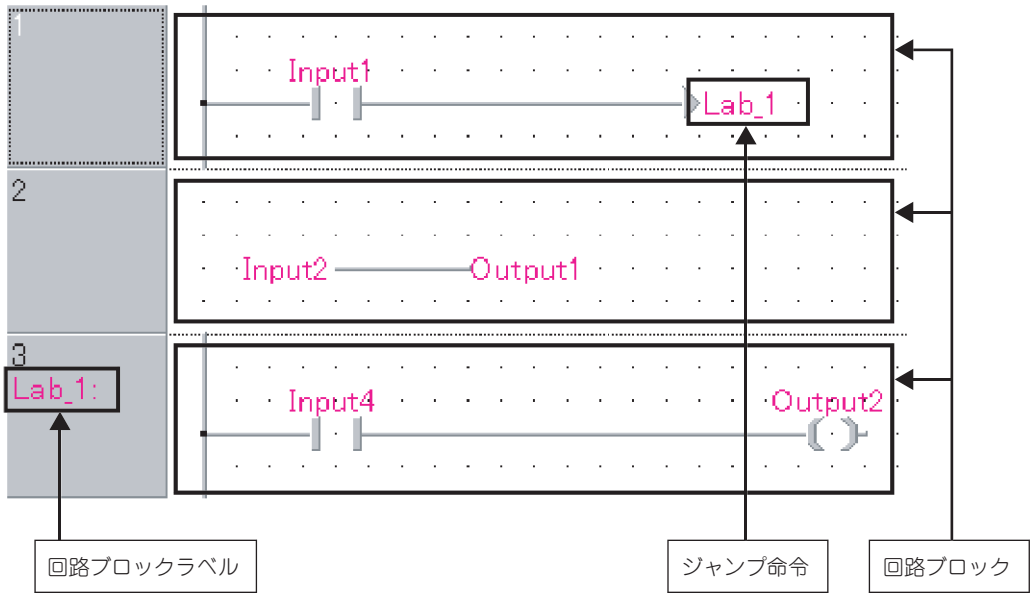
4.2.6 回路ブロック

構造化ラダー /FBD では，プログラムは回路ブロックという単位で区切られます。

ST では，回路ブロックはありません。

● 回路ブロックラベル

回路ブロックには，回路ブロックラベルを付けることができます。ジャンプ命令のジャンプ先として回路ブロックラベルを使用します。



4.2.7 プログラム部品のプログラム言語

プログラム部品のプログラムで，使用できるプログラム言語は 2 種類あります。

下記に，各プログラム言語の特徴を示します。

(1) ST：ストラクチャードテキスト

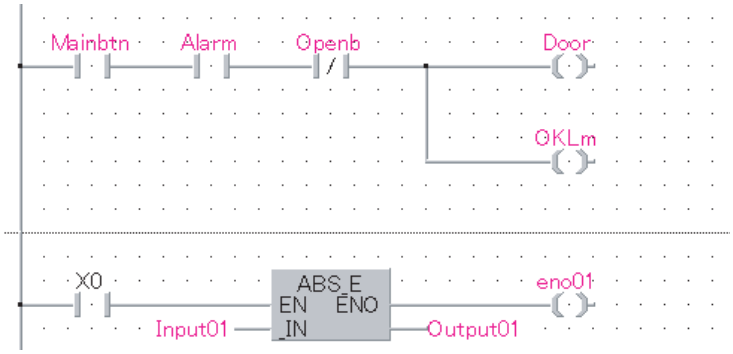
ST は，C 言語などの高級言語と同様に条件文による選択分岐や，反復文による繰り返しなどの構文による制御が記述できます。これにより，見やすいプログラムを簡潔に書くことができます。

```
intV2 := ABS( intV1);

IF M1 THEN
    btn01 := TRUE;
ELSE
    btn01 := FALSE;
END_IF;

Output_ENO := ENEG(btn01, Input1);
```

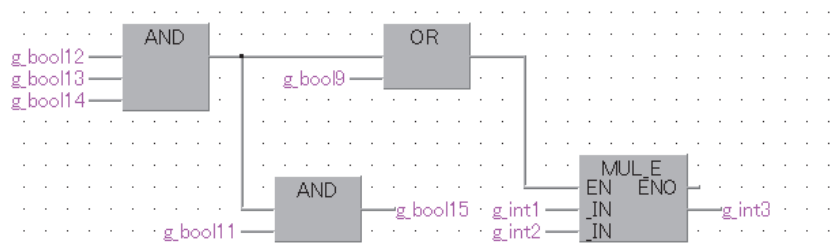
(2) 構造化ラダー /FBD（回路）

構造化ラダー /FBD は，リレー回路の設計技術に基づいて作られたグラフィック言語です。直感的に理解しやすいため，シーケンスプログラムで一般的に用いられています。

- 構造化ラダー



- FBD



4.2.8 ファンクション，ファンクションブロック，オペレータの違い

ファンクション，ファンクションブロック，オペレータの違いを下記に示します。

表 4.2.8-1 ファンクション，ファンクションブロック，オペレータの違い

項目	ファンクション	ファンクションブロック	オペレータ
出力変数の割付け	できません	できます	できません
内部変数	使用しません	使用します	使用しません
インスタンスの作成	不要です	必要です	不要です

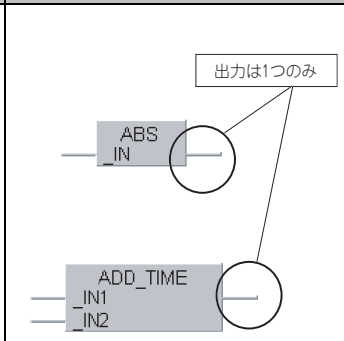
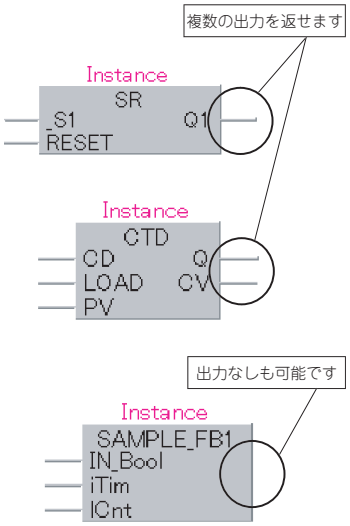
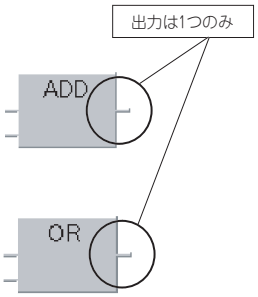
（1）出力変数の割付けについて

ファンクションは，必ず 1 つの演算結果を出力します。出力なし，または複数の演算結果を出力するファンクションは作成できません。

ファンクションブロックは，複数の演算結果を出力することができます。また，出力なしにすることもできます。

オペレータは，必ず 1 つの演算結果を出力します。編集できません。

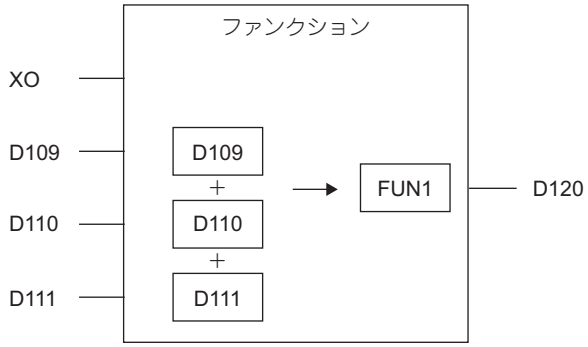
表 4.2.8-2 出力変数の割付け

	ファンクション	ファンクションブロック	オペレータ
例			

(2) 内部変数について

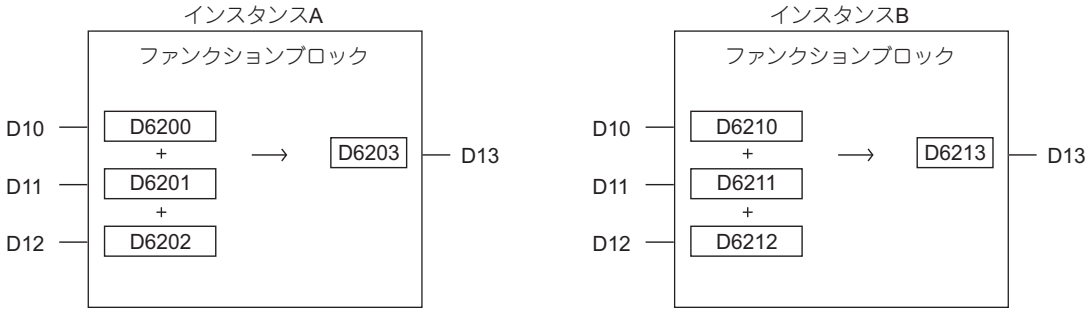
ファンクションは内部変数を使いません。ファンクションは，入力変数ごとに割り付けられた各デバイスをそのまま使用して，演算を繰り返します。

(a) 3 つの入力変数の総和を出力するプログラム（ファンクション (ファンクション名は FUN1) の場合）



ファンクションブロックは内部変数を使います。内部変数には，ファンクションブロックのインスタンスごとに異なるデバイスが割り付けられます。

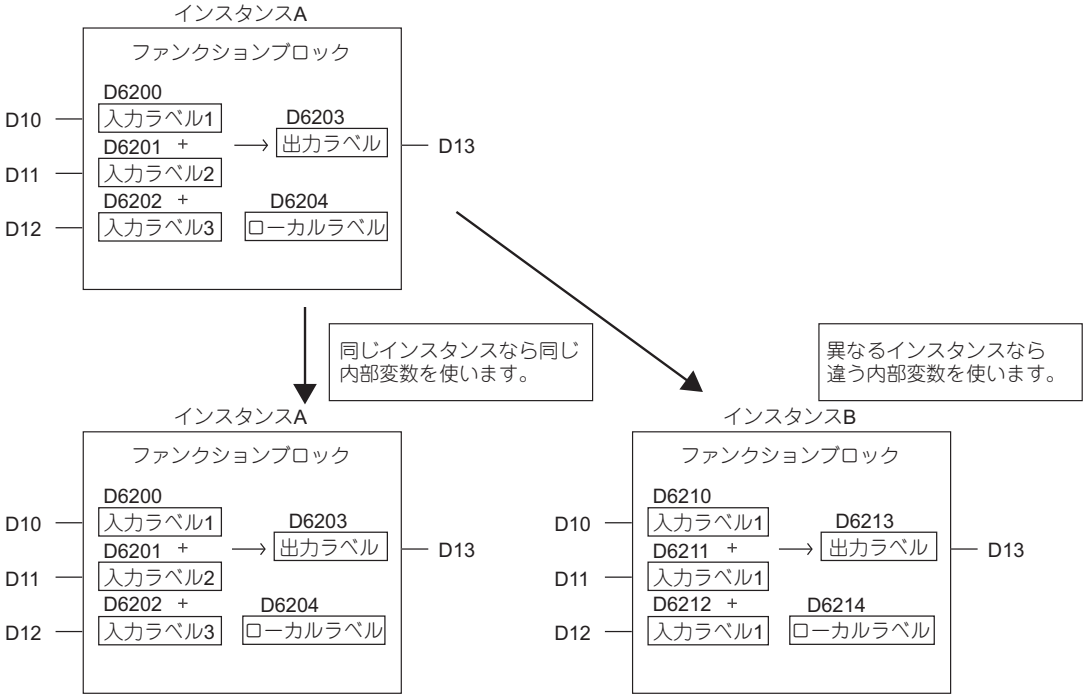
(b) 3 つの入力変数の総和を出力するプログラム（ファンクションブロックの場合）



(3) インスタンスの作成について

ファンクションブロックを使用するときは，内部変数を確保する必要があるため，インスタンスを作成します。
ファンクションブロックのインスタンスを作成することによって，プログラムや他のファンクションブロックから呼び出して，使用できます。

インスタンス化するには，グローバルラベルまたは，ファンクションブロックを使用するプログラム部品のローカルラベルで，ラベルとして宣言します。1 つのプログラム部品の中に，同じファンクションブロックを異なる名前で複数インスタンス化して，使用することができます。



ファンクションブロックは，インスタンスごとに割り当てられた内部変数を使用して，演算を行います。

ポイント

同一ファンクションを回路内で複数回呼び出した場合，ファンクションの内部変数や出力変数の値は呼び出しごとに上書きされます。呼び出しごとに内部変数や出力変数を保持したい場合は，ファンクションブロックを使用する，または出力変数を異なる変数に退避するなどプログラミングしてください。

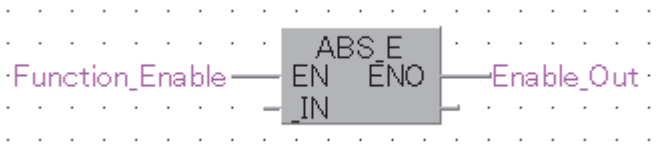
4.2.9 EN/ENO の考え方

ファンクション，ファンクションブロックは EN（イネーブル入力），ENO（イネーブル出力）を付けることにより，実行制御をすることができます。

EN には，ファンクションの実行条件とするブール変数を設定します。

EN 付きファンクションは，EN の実行条件が TRUE の場合のみ，実行します。

ENO には，ファンクションの実行結果を出力するブール変数を設定します。



EN の状態による ENO と演算結果の内容を下表に示します。

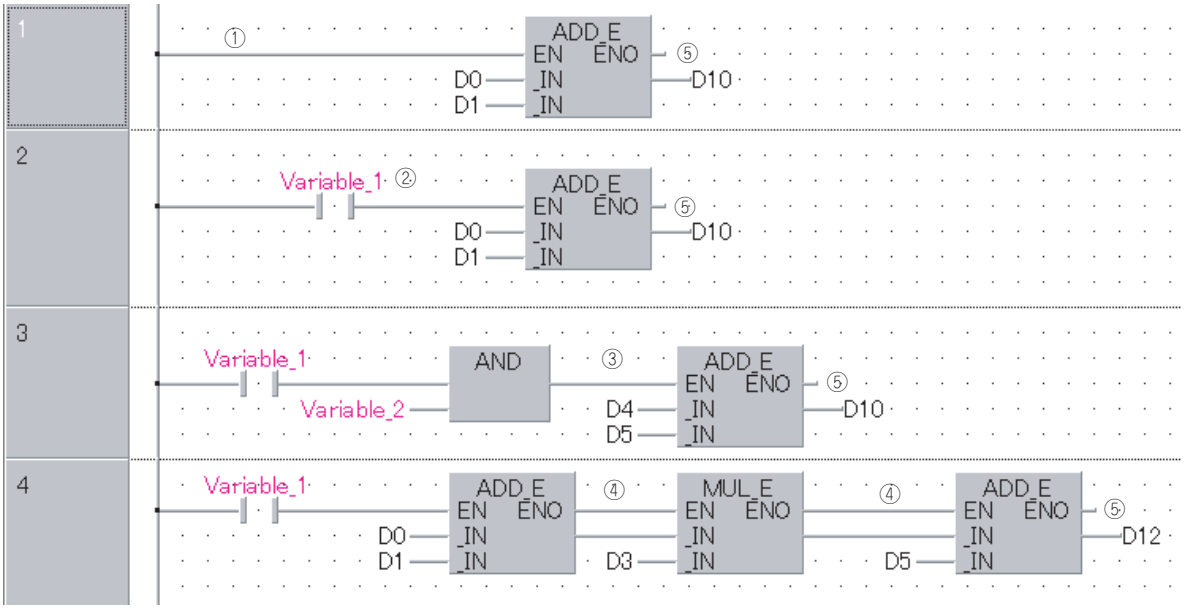
表 4.2.9-1 EN の状態による ENO と演算結果

EN	ENO	演算結果
TRUE（演算実行）	TRUE（演算エラーなし）	演算出力値
	FALSE（演算エラーあり）	不定値
FALSE（演算停止）	FALSE	不定値

ポイント

1. ENO への出力ラベルの設定は，必須ではありません。
2. 応用関数の場合，EN 付き関数は “関数名 _E” となります。

● EN,ENO の使用例



番号	制御内容
①	左側母線から EN 入力へ直接接続する場合は，EN 入力はいつも TRUE であり，命令は常に実行します。ADD_E 命令がこのように使われるとき，EN 入力のない ADD 命令と同じ結果となります。
②	Variable_1 から EN 入力へ接続する場合は，Variable_1 が TRUE のとき命令を実行します。
③	演算のブール結果を EN 入力へ接続する場合は，演算結果が TRUE のとき命令を実行します。
④	ENO 出力から EN 入力へ接続する場合は，Variable_1 が TRUE のとき，3 つの命令を実行します。
⑤	ENO 出力を接続しない場合は，命令の実行結果を出力しません。

4.3 ラベル

ラベルには，グローバルラベルとローカルラベルがあります。

4.3.1 グローバルラベル

グローバルラベルは，プログラムとファンクションブロックで使用できるラベルです。

グローバルラベルの設定では，ラベル名とクラス，データ型およびデバイスの関連付けを行います。

4.3.2 ローカルラベル

ローカルラベルは，宣言したプログラム部品の中でのみ使用できるラベルです。プログラム部品ごとに，個別に定義します。

ローカルラベルの設定では，ラベル名とクラス，データ型の設定を行います。

ローカルラベルでは，ユーザがデバイスを指定する必要はありません。デバイスへの割付けは，コンパイル時に自動的に行われます。

4.3.3 ラベルのクラス

ラベルのクラスは，ラベルがどのプログラム部品から，どのように使用できるかを表します。
クラスは，プログラム部品の種類によって，選択できるものが異なります。

下表に，ラベルのクラスを示します。

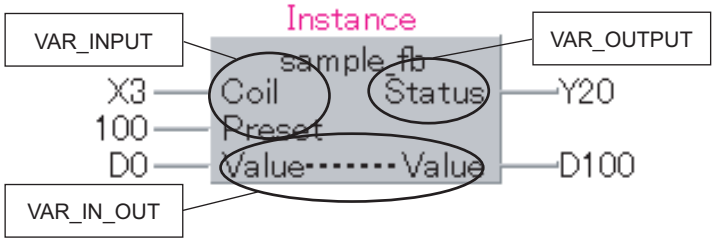
表 4.3.3-1 ラベルのクラス

クラス	内容	使用できるプログラム部品		
		プログラム	ファンクション	ファンクションブロック
VAR_GLOBAL	プログラムとファンクションブロックで使用できる共通のラベルです。	○	×	○
VAR_GLOBAL_CONSTANT	プログラムとファンクションブロックで使用できる共通の定数です。	○	×	○
VAR	宣言したプログラム部品の範囲内で使用するラベルです。他のプログラム部品では使用できません。	○	○	○
VAR_CONSTANT	宣言したプログラム部品の範囲内で使用する定数です。他のプログラム部品では使用できません。	○	○	○
VAR_RETAIN*1	宣言したプログラム部品の範囲内で使用するラッチ型のラベルです。他のプログラム部品では使用できません。	○	×	○
VAR_INPUT	値を受け取るラベルで，プログラム部品内で変更できません。	×	○	○
VAR_OUTPUT	ファンクションブロックから出力するラベルです。	×	×	○
VAR_IN_OUT	値を受け取り，プログラム部品から出力するローカルラベルです。プログラム部品内で変更できます。	×	×	○

*1: FXCPU は対応していません。

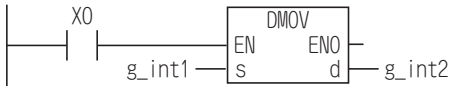
ポイント

・入力変数，出力変数，入出力変数について
VAR_INPUT は，ファンクションやファンクションブロックの入力になります。
VAR_OUTPUT は，ファンクションブロックの出力になります。
VAR_IN_OUT は，入力と出力の両方の役割を果たします。



4.3.4 ラベルの設定

プログラムで使用するラベルは，グローバルラベルまたはローカルラベルの設定が必要です。
下記に DMOV 命令の引数 g_int1,g_int2 の設定例を示します。



- DMOV 命令の引数をグローバルラベルとして使用する場合
  クラス，ラベル名，データ型，デバイス／アドレスを設定します。

	クラス	ラベル名	データ型		定数値	デバイス	アドレス	コメント
1	VAR_GLOBAL	g_int1	ワード[符号付き]	...		D0	%MW0.0	
2	VAR_GLOBAL	g_int2	ワード[符号付き]	...		D10	%MW0.10	
3				...				

- DMOV 命令の引数をローカルラベルとして使用する場合
  クラス，ラベル名，データ型を設定します。

	クラス	ラベル名	データ型		定数値	デバイス	アドレス	コメント
1	VAR	g_int1	ワード[符号付き]	...				
2	VAR	g_int2	ワード[符号付き]	...				
3				...				

4.3.5 データ型

ラベルは，ビット長，処理方法，値の範囲などによって，データ型に分類されています。

(1) 基本データ型

基本データ型には，下記に示す性質を持ったデータ型があります。*1

- ブール型（ビット）：ON か OFF などのような二者択一の状態を表す型です。
- ビット列型(ワード [符号なし]/ ビット列 [16 ビット], ダブルワード [符号なし]/ ビット列 [32 ビット])：ビットの配列を表す型です。
- 整数型(ワード [符号付き], ダブルワード [符号付き])：正および負の整数値を取り扱う型です。
- 実数型（単精度実数，倍精度実数）：小数点以下の数値を取り扱う型です。
- 文字列型（文字列）：文字列（キャラクタ）を取り扱う型です。
- 時間型（時間）：日時分秒（ミリ秒）として数値を取り扱う型です。

表 4.3.5-1 基本データ型

基本データ型	内容	値の範囲	ビット長
ビット	ブール	0(FALSE)，1(TRUE)	1 ビット
ワード [符号付き]	整数	− 32768 ～ 32767	16 ビット
ダブルワード [符号付き]	倍精度整数	− 2147483648 ～ 2147483647	32 ビット
ワード [符号なし]/ ビット列 [16 ビット]	16 ビット列	0 ～ 65535	16 ビット
ダブルワード [符号なし]/ ビット列 [32 ビット]	32 ビット列	0 ～ 4294967295	32 ビット
単精度実数 *2	実数	-2^{128} ～ -2^{-126}，0，2^{-126} ～ 2^{128}	32 ビット
倍精度実数 *3	倍精度実数	-2^{1024} ～ -2^{-1022}，0，2^{-1022} ～ 2^{1024}	64 ビット
文字列 *4	文字列	最大 255 文字	可変
時間 *5	時間値	T#-24d20h31m23s648ms ～ T#24d20h31m23s647ms	32 ビット

*1: 下記に示すデータ型は，構造化ラダー /FBD/ST では使用できません。
ラダーでのみ使用できます。
- タイマ型：シーケンサ CPU のタイマデバイス（T）を取り扱う型です。
- 積算タイマ型：シーケンサ CPU の積算タイマデバイス（ST）を取り扱う型です。
- カウンタ型：シーケンサ CPU のカウンタデバイス（C）を取り扱う型です。
- ポインタ型：シーケンサ CPU のポインタデバイス（P）を取り扱う型です。

*2: FXCPU の場合，FX_{3S}, FX_{3G}, FX_{3GC}, FX_{3U}, FX_{3UC} で使用できます。

*3: ユニバーサルモデル QCPU/LCPU でのみ使用できます。

*4: FXCPU の場合，FX_{3U}, FX_{3UC} で使用できます。

*5: 時間型は，応用関数の時間型演算命令で使用します。応用関数については下記を参照してください。
☞ MELSEC-Q/L 構造化プログラミングマニュアル (応用関数編)
☞ FXCPU 構造化プログラミングマニュアル [応用関数編]

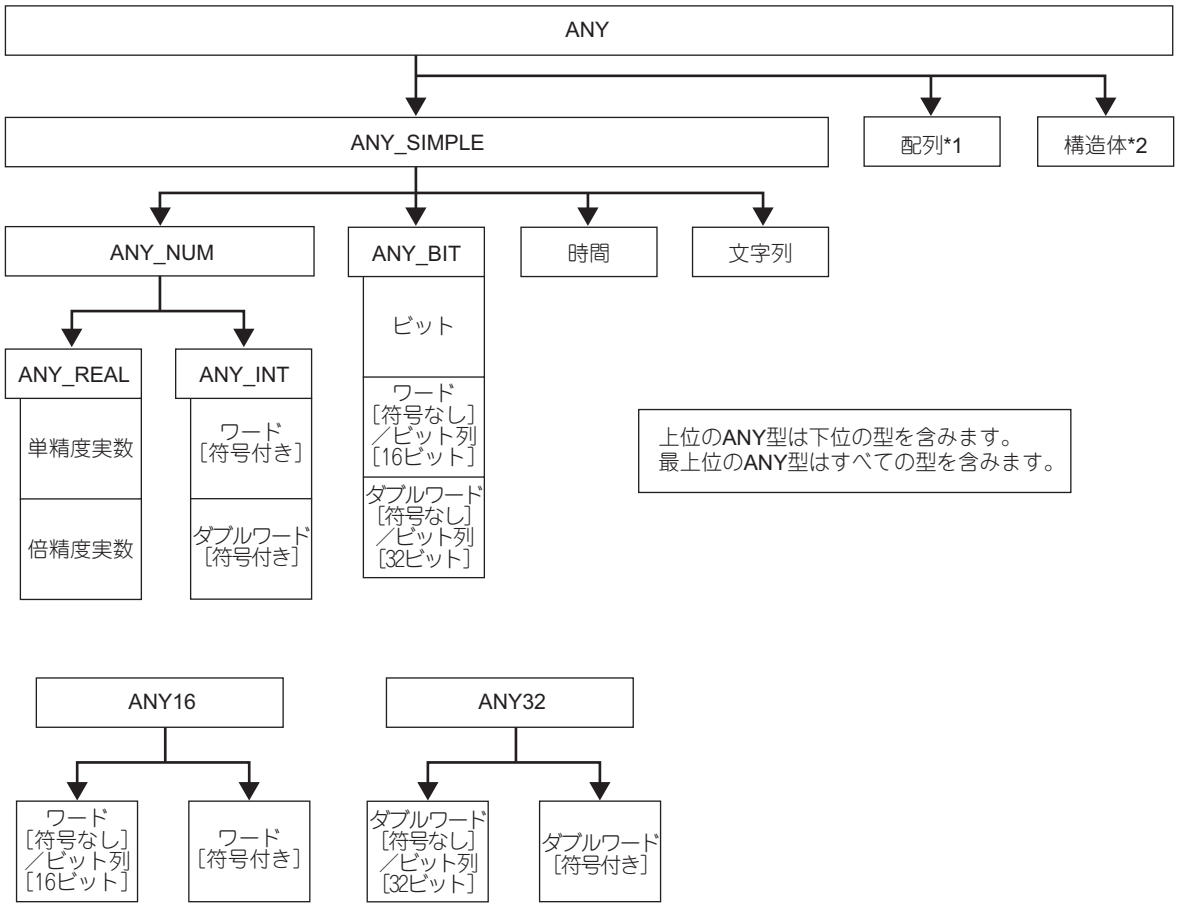
(2) 普遍的データ型

普遍的データ型とは，いくつかの基本データ型をまとめたラベルのデータ型です。データ型名は「ANY」で始まります。

関数の引数，戻り値などで複数のデータ型が許される場合に，ANY 型が使用されています。
普遍的データ型で定義されたラベルは，下位のデータ型のどの型でも使用することができます。
例えば，関数の引数が ANY_NUM の場合には，引数としてワード [符号付き] 型，ダブルワード [符号付き] 型，単精度実数型，倍精度実数型から任意のデータ型を指定できます。

関数や命令の引数は，各種の異なる型のデータにも利用できるように普遍的データ型を使って記述されています。

下記に普遍的データ型の種類と，対応する基本データ型を示します。



*1: 配列は ☞ 4.4.7 項 配列

*2: 構造体は ☞ 4.4.8 項 構造体

4.3.6 定数の表記方法

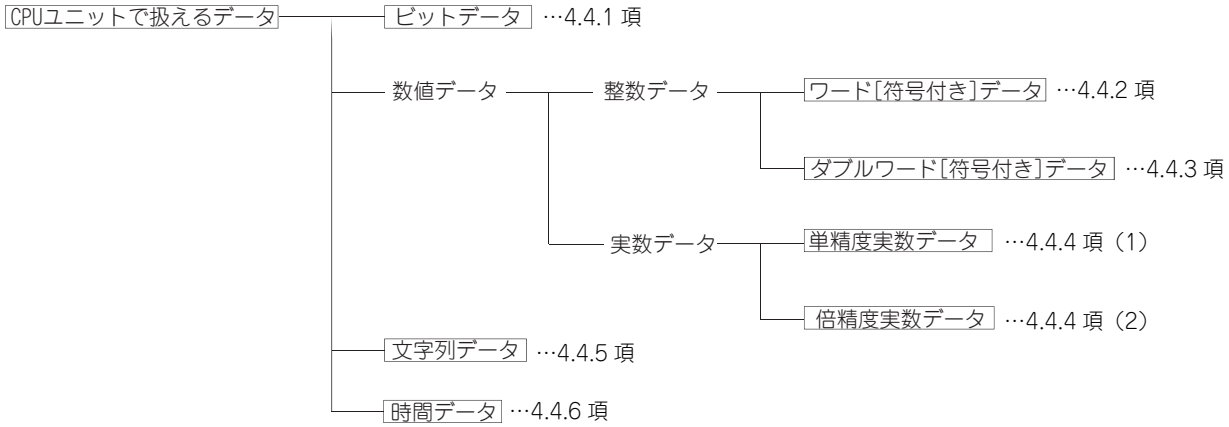
ラベルに定数を設定するときの表記方法は，下記のとおりです。

表 4.3.6-1 定数の表記方法

定数の種類	表記方法	例
ブール	FALSE か TRUE で入力します。または 0 か 1 で入力します。	TRUE，FALSE
2 進数	2 進数の前に “2#” をつけます。	2#0010， 2#01101010
8 進数	8 進数の前に “8#” をつけます。	8#0，8#337
10 進数	10 進数を直接入力します。または 10 進数の前に “K” をつけます。	123，K123
16 進数	16 進数の前に “16#” をつけます。または “H” をつけます。 “H” の場合，アルファベットは自動で大文字に変換されます。	16#FF，HFF
実数	実数を直接入力します。または実数の前に “E” をつけます。	2.34，E2.34
文字列	文字列をシングルクォーテーション (') またはダブルクォーテーション (") で囲みます。	'ABC'，"ABC"
時間	先頭に “T#” をつけます。	T#1 h， T#1d2h3m4s5ms

4.4 データの指定方法

CPU ユニットの命令で使用できるデータには，下記 6 種類があります。



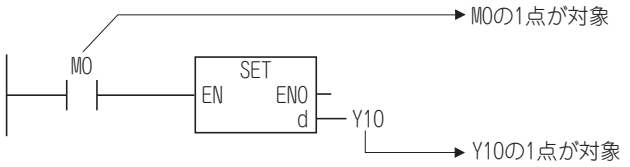
4.4.1 ビットデータの場合

ビットデータは，接点・コイルなど 1 ビット単位で扱うデータです。

ビットデータとして，“ビットデバイス” および “ビット指定したワードデバイス” が使用できます。

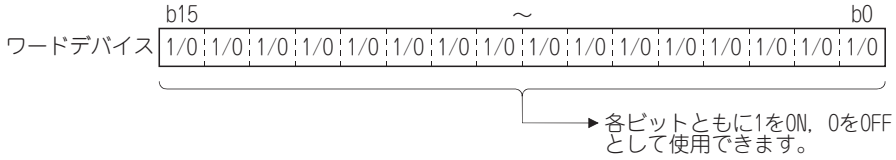
(1) ビットデバイス使用時

ビットデバイスは，1 点単位で指定します。

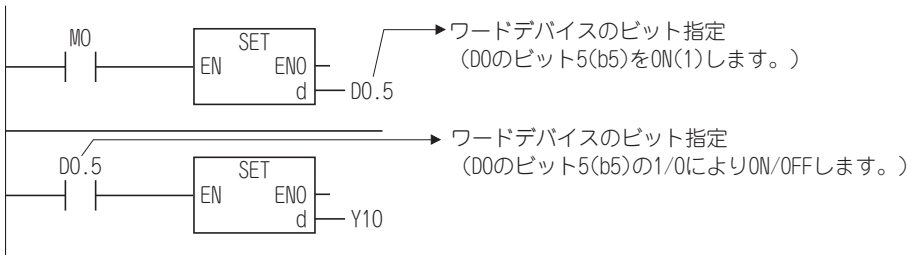


(2) ワードデバイス使用時

(a) ワードデバイスはビット No. 指定を行うことにより，指定ビット No. の 1/0 をビットデータとして使用できます。



(b) ワードデバイスのビット指定は，“ワードデバイス.ビット No.” で指定します。
(ビット No. の指定は，16 進数です。)
たとえば D0 のビット 5(b5) は D0.5，D0 のビット 10(b10) は D0.A で指定します。
ただしタイマ (T)，積算タイマ (ST)，カウンタ (C)，インデックスレジスタ (Z) は，ビット指定できません。(例：Z0.0 は不可)



ポイント

FXCPU の場合，ワードデバイスのビット指定は FX3U，FX3UC で使用できます。

4.4.2 ワード（16 ビット）データの場合

ワードデータは，基本命令・応用命令で使用する 16 ビットの数値データです。

CPU ユニットで扱えるワードデータは，下記 2 種類です。

・ 10 進定数 ····· -32768 ～ 32767
・ 16 進定数 ····· 0000H ～ FFFFH

ワードデータは，ワードデバイスおよび桁指定したビットデバイスが使用できます。

ただしダイレクトアクセス入力（DX），ダイレクトアクセス出力（DY）は桁指定によるワードデータ指定はできません。（ダイレクトアクセス入力，ダイレクトアクセス出力は，使用している CPU ユニットのユーザーズマニュアル（機能解説・プログラム基礎編）を参照してください。）

(1) ビットデバイス使用時

(a) ビットデバイスは桁指定により，ワードデータを扱うことができます。

ビットデータの桁指定は，“桁数 ビットデバイスの先頭番号” で指定します。
桁指定は 4 点（4 ビット）単位で，K1 ～ K4 まで指定できます。
（リンクダイレクトデバイスの場合は，
“J ネットワークNo. ¥ 桁数 ビットデバイスの先頭番号” で指定します。ネットワーク No.2 の X100 ～ X10F を指定する場合は，J2¥K4X100 になります。）
たとえば X0 を桁指定した場合の対象点数は，次のようになります。

QCPU(Q モード)/LCPU の場合

- K1X0 ···· X0 ～ X3 の 4 点が対象
- K2X0 ···· X0 ～ X7 の 8 点が対象
- K3X0 ···· X0 ～ XB の 12 点が対象
- K4X0 ···· X0 ～ XF の 16 点が対象

FXCPU の場合
（FXCPU では，入出力（X,Y）のデバイス番号を 8 進数で割り付けています。）

- K1X0 ···· X0 ～ X3 の 4 点が対象
- K2X0 ···· X0 ～ X7 の 8 点が対象
- K3X0 ···· X0 ～ X13 の 12 点が対象
- K4X0 ···· X0 ～ X17 の 16 点が対象

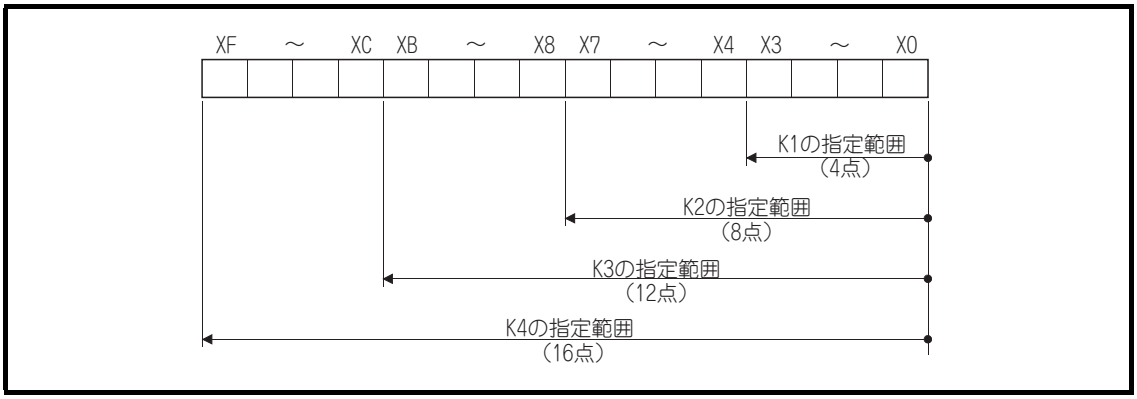


図 4.4.2-1 ワードデータ（16 ビット ）の桁指定設定範囲

(b) ソース ⓢ 側に桁指定のある場合，ソースデータとして取り扱える数値は表 4.4.2-1 のようになります。

表 4.4.2-1 桁指定と取り扱える数値の一覧

指定桁数	値の範囲
K1（4 点）	0 ～ 15
K2（8 点）	0 ～ 255
K3（12 点）	0 ～ 4095
K4（16 点）	–32768 ～ 32767

(c) ディスティネーションがワードデバイスの場合
ディスティネーション側のワードデバイスは，ソース側で桁指定したビット以降のビット状態には 0 が格納されます。

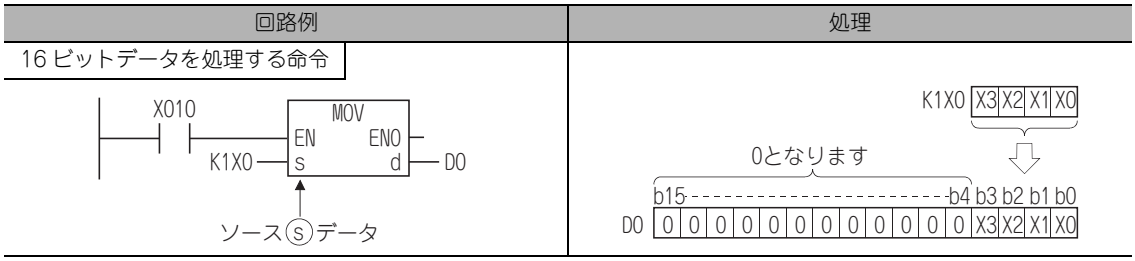


図 4.4.2-2 回路例と処理内容

(d) ディスティネーションⓓ側に桁指定のある場合は，桁指定による点数がディスティネーション側の対象となります。
桁指定による点数以降のビットデバイスは，変化しません。

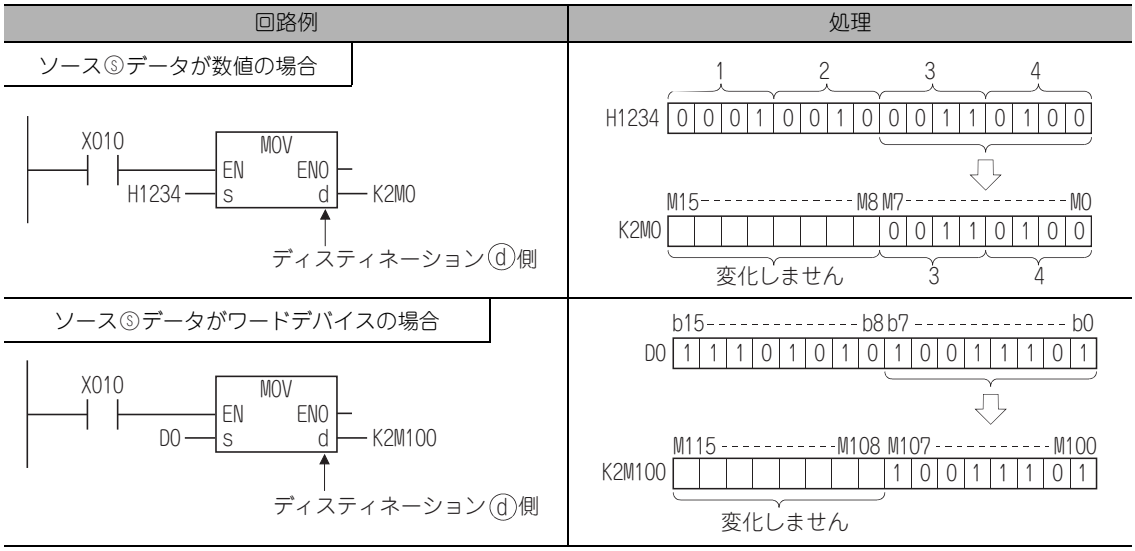
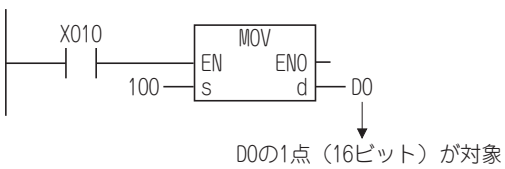


図 4.4.2-3 回路例と処理内容

(2) ワードデバイス使用時
ワードデバイスは，1 点（16 ビット）単位で指定します。



ポイント

1. 桁指定処理を行う場合，ビットデバイスの先頭デバイス No. は任意の値が使用できます。
2. ダイレクトアクセス入出力 (DX,DY) は桁指定できません。

4.4.3 ダブルワード（32 ビット）データの場合

ダブルワードデータは，基本命令・応用命令で使用する 32 ビットの数値データです。
CPU ユニットで扱えるダブルワードデータは，下記 2 種類です。
・ 10 進定数 ····· -2147483648 ～ 2147483647
・ 16 進定数 ····· 00000000H ～ FFFFFFFFH
ダブルワードデータは，ワードデバイスおよび桁指定したビットデバイスが使用できます。
ただしダイレクトアクセス入力 (DX)，ダイレクトアクセス出力 (DY) は，桁指定によるダブルワードデータ指定はできません。

(1) ビットデバイス使用時

(a) ビットデバイスは桁指定により，ダブルワードデータを扱うことができます。
ビットデータの桁指定は，“桁数 ビットデバイスの先頭番号” で指定します。
（リンクダイレクトデバイスの場合は，
“J ネットワークNo. ¥ 桁数 ビットデバイスの先頭番号 ” で指定します。ネットワーク No.2 の X100 ～ X11F を指定する場合は，J2¥K8X100 になります。）
桁指定は 4 点（4 ビット）単位で，K1 ～ K8 まで指定できます。
たとえば X0 を桁指定した場合の対象点数は，次のようになります。

QCPU(Q モード)/LCPU の場合

- K1X0 ···· X0 ～ X3 の 4 点が対象
- K2X0 ···· X0 ～ X7 の 8 点が対象
- K3X0 ···· X0 ～ XB の 12 点が対象
- K4X0 ···· X0 ～ XF の 16 点が対象
- K5X0 ····X0 ～ X13 の 20 点が対象
- K6X0 ····X0 ～ X17 の 24 点が対象
- K7X0 ····X0 ～ X1B の 28 点が対象
- K8X0 ····X0 ～ X1F の 32 点が対象

FXCPU の場合
（FXCPU では，入出力（X,Y）のデバイス番号を 8 進数で割り付けています。）

- K1X0 ···· X0 ～ X3 の 4 点が対象
- K2X0 ···· X0 ～ X7 の 8 点が対象
- K3X0 ···· X0 ～ X13 の 12 点が対象
- K4X0 ···· X0 ～ X17 の 16 点が対象
- K5X0 ····X0 ～ X23 の 20 点が対象
- K6X0 ····X0 ～ X27 の 24 点が対象
- K7X0 ····X0 ～ X33 の 28 点が対象
- K8X0 ····X0 ～ X37 の 32 点が対象

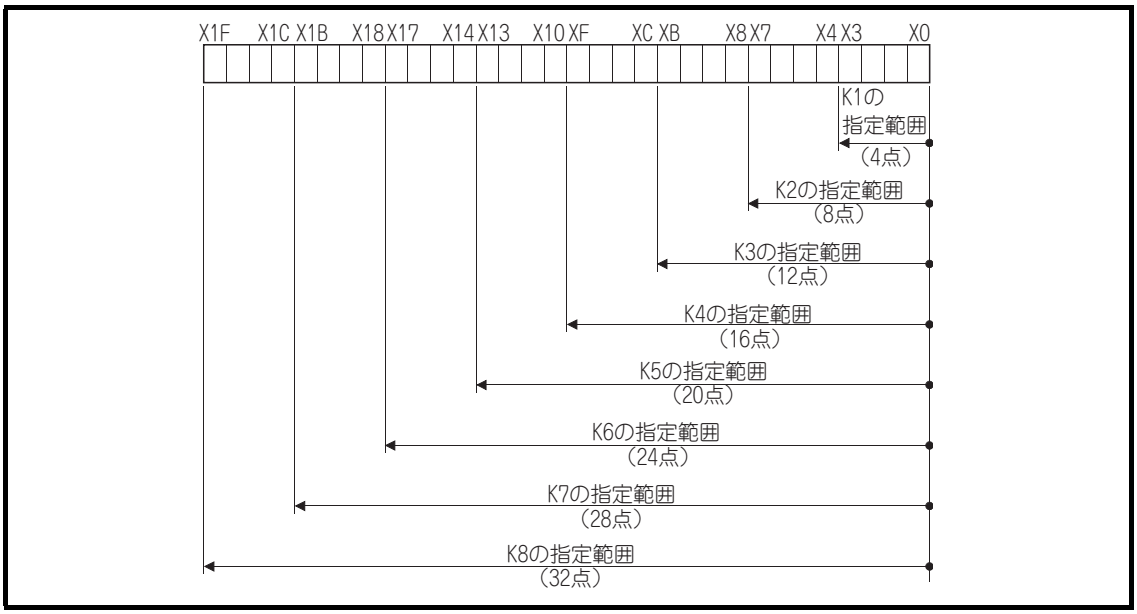


図 4.4.3-1 ダブルワードデータ (32 ビット) の桁指定設定範囲

(b) ソース ⓢ 側に桁指定のある場合，ソースデータとして取り扱える数値は表 4.4.3-1 のようになります。

表 4.4.3-1 桁指定と取り扱える数値の一覧

指定桁数	値の範囲	指定桁数	値の範囲
K1（4 点）	0 ～ 15	K5（20 点）	0 ～ 1048575
K2（8 点）	0 ～ 255	K6（24 点）	0 ～ 16777215
K3（12 点）	0 ～ 4095	K7（28 点）	0 ～ 268435455
K4（16 点）	0 ～ 65535	K8（32 点）	−2147483648 ～ 2147483647

(c) ディスティネーションがワードデバイスの場合
ディスティネーション側のワードデバイスは，ソース側で桁指定したビット以降のビット状態には 0 が格納されます。（Data_s:K1X0, Data_d:D0 の場合）

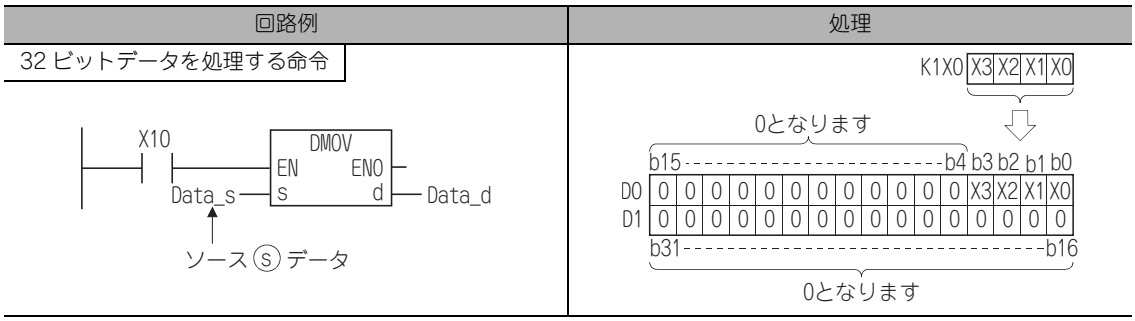


図 4.4.3-2 回路例と処理内容

(d) ディスティネーション ⓓ 側に桁指定のある場合は，桁指定による点数がディスティネーション側の対象となります。（Data_d1:K5M0, Data_d2:K5M10, Data_s:D0 の場合）桁指定による点数以降のビットデバイスは，変化しません。

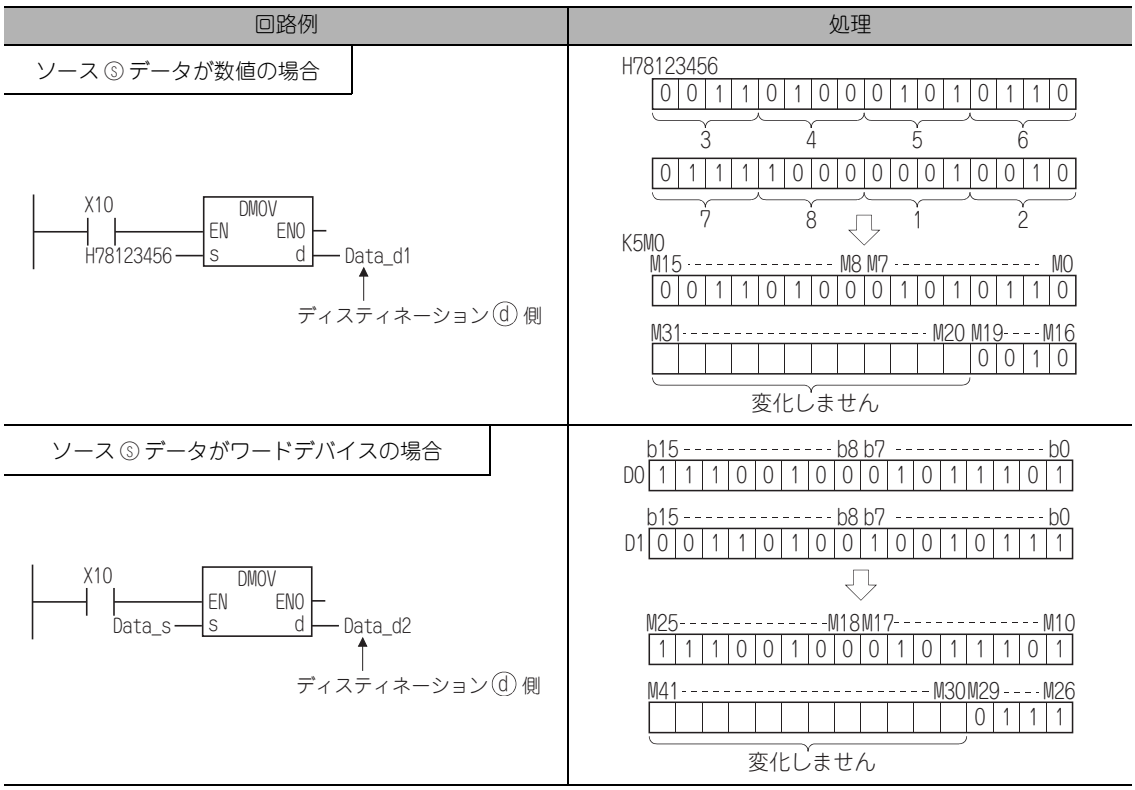


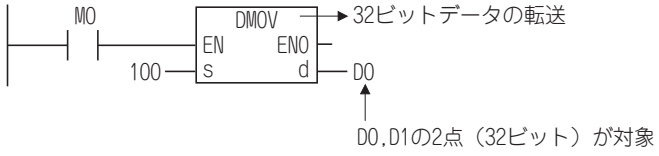
図 4.4.3-3 回路例と処理内容

ポイント

1. 桁指定処理を行う場合，ビットデバイスの先頭デバイス No. は任意の値が使用できます。
2. ダイレクトアクセス入出力 (DX,DY) は桁指定できません。

(2) ワードデバイス使用時

ワードデバイスは，下位 16 ビットで使用するデバイスを指定します。
32 ビットデータを処理する命令では，（指定デバイス番号）と（指定デバイス番号 +1）を使用します。



4.4.4 単精度実数／倍精度実数データの場合

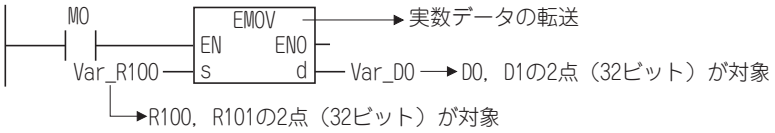
基本命令，応用命令で使用する 32 ビットの浮動小数点データです。

FXCPU の場合，倍精度実数データは対応していません。

ワードデバイスのみ実数データを格納できます。

(1) 単精度実数（単精度浮動小数点）

実数を扱う命令では，下位 16 ビットで使用するデバイスを指定します。
実数は，（指定デバイス番号）と（指定デバイス番号 +1）の 32 ビットに格納されます。

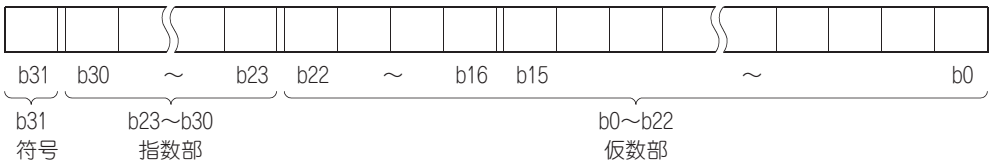


備 考

浮動小数点データはワードデバイス 2 つを使って次のように表します。

[符号] 1.[仮数部]×2 [指数部]

浮動小数点データを内部表現するときのビット構成とその意味は次のとおりです。



- 符号　b31 にて符号を表します。
  0：正
  1：負

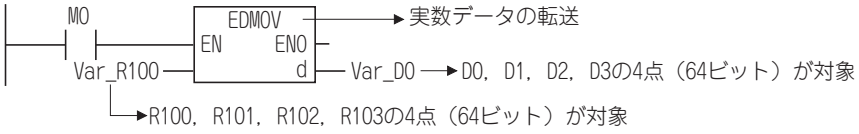
- 指数部　b23 ～ b30 にて 2^n の n を表現します。
  b23 ～ b30 の BIN 値により n は次のようになります。

b23～b30	FFH	FEH	FDH		81H	80H	7FH	7EH		02H	01H	00H
n	未使用	127	126		2	1	0	−1		−125	−126	未使用

- 仮数部　b0～b22の23ビットにて，2進数で1.XXXXXX…としたときのXXXXXX… の値を示します。

(2) 倍精度実数（倍精度浮動小数点）

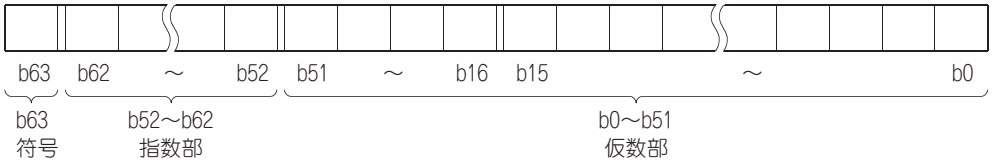
実数を扱う命令では，下位 16 ビットで使用するデバイスを指定します。
実数は，（指定デバイス番号）と（指定デバイス番号＋ 3）の 64 ビットに格納されます。



備 考

浮動小数点データはワードデバイス 4 つを使って次のように表します。
[符号] 1.[仮数部]×2 [指数部]

浮動小数点データを内部表現するときのビット構成とその意味は次のとおりです。



- 符号　b63 にて符号を表します。
  0：正
  1：負

- 指数部　b52 ～ b62 にて 2^n の n を表現します。
  b52 ～ b62 の BIN 値により n は次のようになります。

b52～b62	7FFH	7FEH	7FDH			400H	3FFH	3FEH	3FDH	3FCH			02H	01H	00H
n	未使用	1023	1022			2	1	0	−1	−2			−1021	−1022	未使用

- 仮数部　b0～b51 の 52 ビットにて，2 進数で 1.XXXXXX… としたときの XXXXXX… の値を示します。

(3) プログラミングツールから単精度実数／倍精度実数の入力値を設定する場合の注意事項

(a) 単精度実数

プログラミングツールでは，単精度実数型データは 32 ビットの単精度で処理するため，有効桁数は約 7 桁となります。したがって単精度実数型データの入力値が 7 桁を超えた場合，8 桁目を四捨五入した値に丸めます。
そのため，四捨五入後の値が -2147483648 ～ 2147483647 を超えると，オペレーションエラーが発生します。

例1）入力値に「2147483647」を設定した場合

8桁目の「6」を四捨五入するため，
「2147484000」として扱われます。

例2）入力値に「E1.1754943562」を設定した場合

8桁目の「3」を四捨五入するため，
「E1.175494」として扱われます。

(b) 倍精度実数

プログラミングツールでは，倍精度実数型データは 64 ビットの倍精度で処理するため，有効桁数は約 15 桁となります。したがって倍精度実数型データの入力値が 15 桁を超えた場合，16 桁目を四捨五入した値に丸めます。
そのため，四捨五入後の値が -2147483648 ～ 2147483647 を超えると，オペレーションエラーが発生します。

例1）入力値に「2147483646.12345678」を設定した場合

16桁目の「6」を四捨五入するため，
「2147483646.12346」として扱われます。

例2）入力値に「E1.7976931348623157+307」を設定した場合

16桁目の「5」を四捨五入するため，
「E1.79769313486232+307」として扱われます。

ポイント

1. プログラミングツールのモニタ機能では CPU ユニットの浮動小数点データをモニタできます。
2. 浮動小数点データで 0 を表すときは，下記の範囲をすべて 0 にします。
   (a) 単精度浮動小数点データの場合，b0 ～ b31
   (b) 倍精度浮動小数点データの場合，b0 ～ b63
3. 実数の設定範囲は下記のとおりです。*1
   (a) 単精度浮動小数点データの場合
   $-2^{128} < \text{デバイス} \leqq -2^{-126}$，0，$2^{-126} \leqq \text{デバイス} < 2^{128}$
   (b) 倍精度浮動小数点データの場合
   $-2^{1024} \leqq \text{デバイス} \leqq -2^{-1022}$，0，$-2^{-1022} \leqq \text{デバイス} \leqq 2^{1024}$
4. 浮動小数点データで，−0（浮動小数点型実数の最上位ビットのみ 1 のとき）は指定しないでください。（−0 で浮動小数点演算を行うと演算エラーになります。）
   浮動小数点演算の内部演算を倍精度で行っている CPU ユニットは，−0 が指定されると CPU ユニット内部で 0 に変換して，浮動小数点演算を行うため，演算エラーになりません。
   浮動小数点演算の内部演算を単精度で行っている CPU ユニットは，−0 が指定されると処理速度を優先し，−0 をそのまま浮動小数点演算に使用するため演算エラーになります。
   (a) −0 を指定したとき演算エラーにならない CPU ユニットを下記に示します。
   - 内部演算を倍精度に設定しているハイパフォーマンスモデル QCPU *2
     （浮動小数点演算の内部演算のデフォルトは倍精度です。）

   (b) −0 を指定したとき演算エラーになる CPU ユニットを下記に示します。
   - ベーシックモデル QCPU*3
   - 内部演算を単精度に設定しているハイパフォーマンスモデル QCPU *2
   - プロセス CPU
   - 二重化 CPU
   - ユニバーサルモデル QCPU
   - LCPU
   - FXCPU*4

＊1：オーバーフローおよびアンダーフロー時の動作や，特殊な値を入力したときの動作については，下記を参照してください。
・QCPU(Q モード)/LCPU の場合
☞使用している CPU ユニットのユーザーズマニュアル（機能解説・プログラム基礎編）
・FXCPU の場合
☞使用している FXCPU のユーザーズマニュアル，プログラミングマニュアル

＊2：浮動小数点演算の内部演算の単精度と倍精度の切換えは，PC パラメータの PC システム設定にて行います。浮動小数点演算の単精度，倍精度は，使用している CPU ユニットのユーザーズマニュアル（機能解説・プログラム基礎編）を参照してください。

＊3：ベーシックモデル QCPU は，シリアル No. の上 5 桁が “04122 以降 ” で浮動小数点演算が可能です。

＊4：FXCPU では，FX_{2N}, FX_{2NC}, FX_{3S}, FX_{3G}, $FX3_{GC}$, FX_{3U}, $FX3_{UC}$ のみ浮動小数点演算に対応しています。

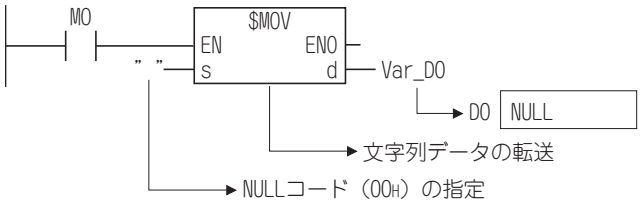
4.4.5 文字列データの場合

基本命令，応用命令で使用する文字データです。

指定文字から文字列の最終を示す NULL コード（00H）までが対象になります。

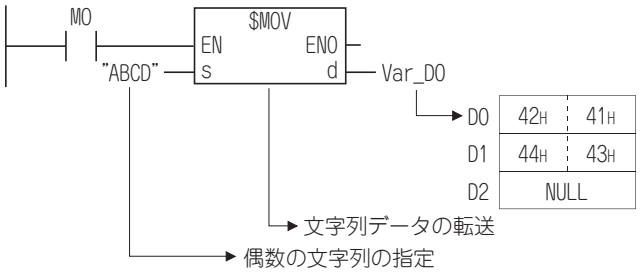
(1) 指定文字が NULL コードの場合

1 ワードを使用して NULL コードを格納します。



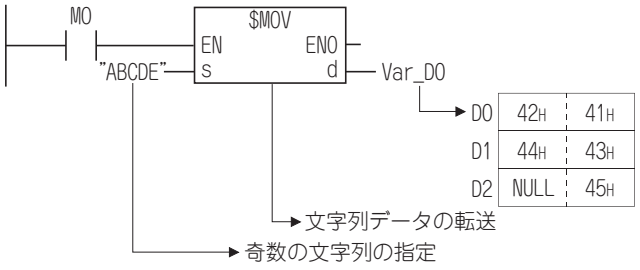
(2) 文字数が偶数の場合

（文字数 /2+1）ワードを使用して，文字列と NULL コードを格納します。
たとえば “ABCD” を D0 ～に転送すると，D0 と D1 に文字列 (ABCD) を格納し，D2 に NULL コードを格納します。(NULL コードは最後の 1 ワードに格納されます。)



(3) 文字数が奇数の場合

（文字数 /2）ワード（小数点以下切り上げ）を使用して，文字列と NULL コードを格納します。
たとえば “ABCDE” を D0 ～に転送すると，D0 ～ D2 に文字列 (ABCDE) と NULL コードを格納します。(NULL コードは最後の 1 ワードの上位 8 ビットに格納されます。)

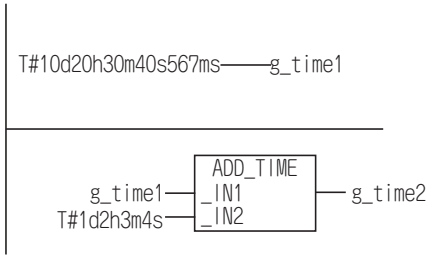


4.4.6 時間データの場合

応用関数の時間型演算命令で使用するデータです。

T#10d20h30m40s567ms の形式で指定します。

たとえば “10 日 20 時 30 分 40 秒 567 ミリ秒” に “1 日 2 時 3 分 4 秒” を加える場合は次のようになります。



時間データの各値は下記の範囲で指定できます。

表 4.4.6-1 時間データの指定可能な範囲

値	範囲
d（日）	0 ～ 24
h（時）	0 ～ 23
m（分）	0 ～ 59
s（秒）	0 ～ 59
ms（ミリ秒）	0 ～ 999

応用関数については下記を参照してください。

☞ MELSEC-Q/L 構造化プログラミングマニュアル (応用関数編)

☞ FXCPU 構造化プログラミングマニュアル [応用関数編]

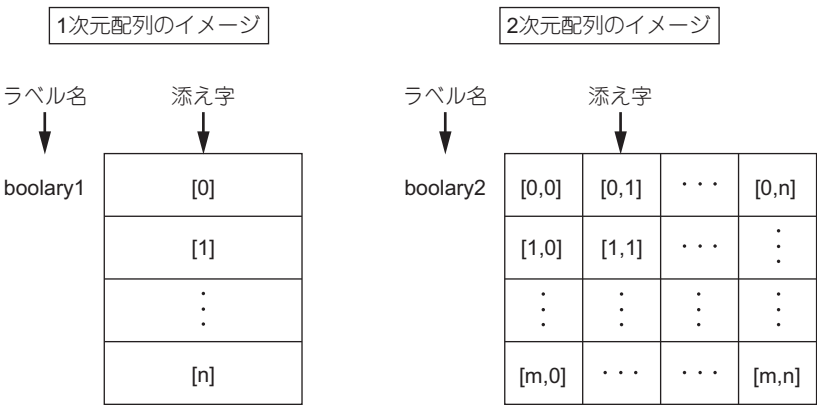
4.4.7 配列

配列は，同じデータ型のラベルの連続的な集合体を，1 つの名前で表したものです。

配列は，基本データ型や構造体で定義できます。

（ ☞ GX Works2 Version 1　オペレーティングマニュアル（構造化プロジェクト編））

データ型によって，配列の最大数が異なります。



（1）配列の定義

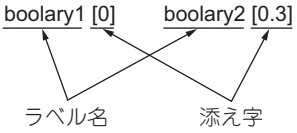
下表に，定義の書式を示します。

表 4.4.7-1 配列を定義するときの書式

配列の次元数	書式	備考
1 次元	基本データ型／構造体名の配列（配列開始値 .. 配列終了値）	基本データ型について ☞ 4.3.5 項 構造体名について ☞ 4.4.8 項
	（定義例） ビット（0..2）	
2 次元	基本データ型／構造体名の配列（配列開始値 .. 配列終了値，配列開始値 .. 配列終了値）	
	（定義例） ビット（0..2，0..1）	
3 次元	基本データ型／構造体名の配列（配列開始値 .. 配列終了値，配列開始値 .. 配列終了値，配列開始値 .. 配列終了値）	
	（定義例） ビット（0..2，0..1，0..3）	

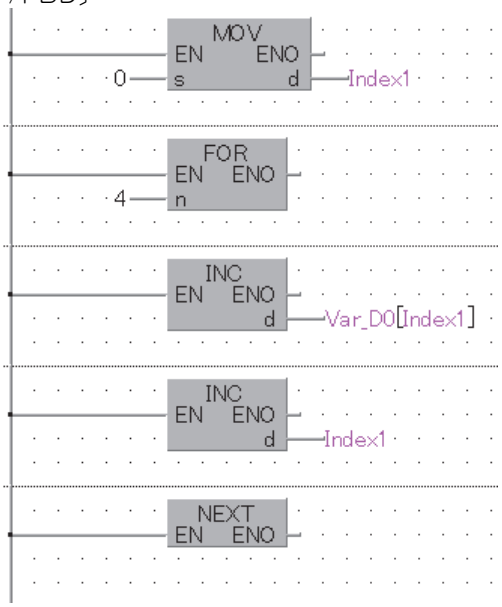
(2) 配列の表記

配列の個々のラベルを識別するために，ラベル名のあとに“[ ]” で添え字を囲んで表します。
添え字に指定できるのは，-32768 ～ 32767 の範囲です。 例）



2 次元以上の配列の場合は，“[ ]” の添え字を “,” で区切って表します。
プログラム言語の ST, 構造化ラダー /FBD では，下記のように，添え字にラベル（ワード [符号付き] 型またはダブルワード [符号付き] 型）を使用することもできます。
ただし，添え字にラベルを使用した場合，プログラム中で Z0，Z1 は使用しないでください。

〔構造化ラダー/FBD〕



〔ST〕

```
FOR Index1:=0
    TO 4
    BY 1 DO
    INC(TRUE,Var_D0[Index1]);
END_FOR;
```

ポイント

(1) 配列の添え字にラベルまたはデバイスを指定した場合，複数のシーケンス命令を組み合わせて演算を行います。そのため，配列ラベルの演算中に割込みが発生すると，意図しない演算結果になる場合があります。
割込みプログラムを使用する場合は，必要に応じて割込禁止／許可命令 (DI/EI 命令) を使用してください。

(2) 配列の添え字に，定義されている範囲外の添え字 *1 を指定すると，下記のような動作となります。
・オペレーションエラーが発生する。
・他のラベルの現在値を参照または書き込みをする。
*1:（例）ビット配列（0..2）で宣言された配列の添え字に，0 ～ 2 以外の数字を使用した場合。

(3) 配列要素数の上限

配列の最大数はデータ型によって異なり、下表のようになります。

表 4.4.7-2 配列の最大数

データ型	最大数
ビット，ワード［符号付き］，ワード［符号なし］／ビット列［16 ビット］， タイマ，カウンタ，積算タイマ	32768
ダブルワード［符号付き］，ダブルワード［符号なし］／ビット列［32 ビット］， 単精度実数，時間	16384
倍精度実数	8192
文字列	32768÷ 文字列長

4.4.8 構造体

構造体は，異なるデータ型のラベルの集合体です。

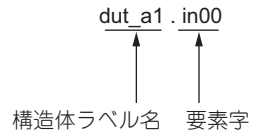
すべてのプログラム部品で使用することができます。

構造体を使用するときは，まず構造体の構成を作成し，次に作成した構造体を新しいデータ型として構造体ラベル名を定義することが必要です。

（☞ GX Works2 Version 1　オペレーティングマニュアル（構造化プロジェクト編））

構造体を構成する個々の要素を使用するには，構造体ラベル名のあとに “.” で区切って要素名をつけます。

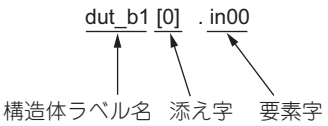
例）構造体の要素を使用する場合

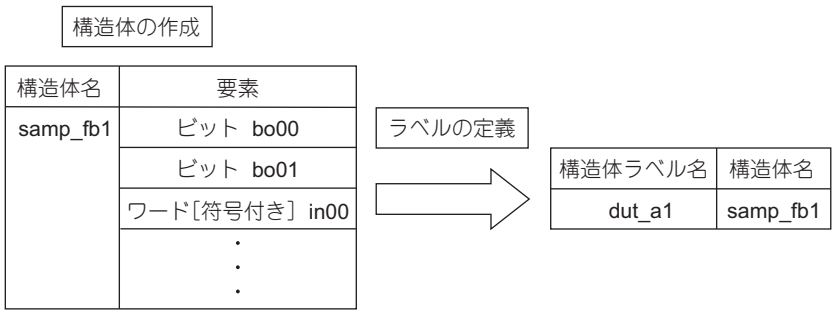


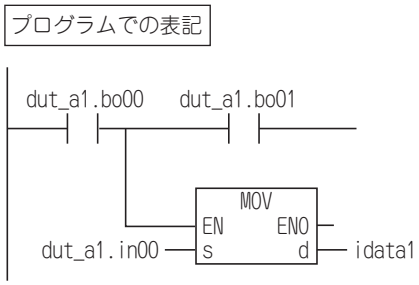
構造体を配列にして使用することも可能です。配列として宣言した場合は，構造体ラベル名のあとに “[ ]” で添え字を囲んで表します。構造体を配列にし，添え字にラベル／デバイスを指定してアクセスする場合の最大データ数は 32767 です。

構造体の配列を，ファンクションやファンクションブロックの引数として指定することもできます。構造体を配列にし、添え字にラベル／デバイスを指定してアクセスする場合、ビット型要素にワードデバイスのビット指定はできません。

例）配列にした構造体の要素を使用する場合







4.5 デバイスとアドレス

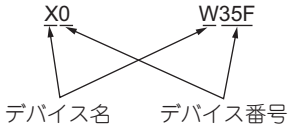
シーケンサ CPU のデバイスの表記方法には，下記の 2 種類があります。

・ デバイス：デバイス名とデバイス番号からなります。

・ アドレス：IEC61131-3 での表記方法です。%で始まります。

4.5.1 デバイス

例）



デバイスは，デバイス名とデバイス番号で表記する方法です。

デバイスの詳細は下記を参照してください。

☞ 使用している CPU ユニットのユーザーズマニュアル（機能解説・プログラム基礎編）

☞ FXCPU 構造化プログラミングマニュアル [デバイス・共通説明編]

4.5.2 アドレス

アドレスは，IEC61131-3 で定義されている表記方法です。

IEC61131-3 にしたがって下記のような表記方法となっています。

表 4.5.2-1 アドレス定義仕様

先頭	1 文字目：位置		2 文字目：サイズ		3 文字目以降：区分	番号
%	I	入力	（省略）	ビット	詳細分類のための数字です。後続の番号とは “.” （ピリオド）で区切って表します。省略されることもあります。	デバイス番号にあたる番号です（10 進数表記です）。
	Q	出力	X	ビット		
	M	内部	W	ワード（16 ビット）		
			D	ダブルワード（32 ビット）		
			L	ロングワード（64 ビット）*1		

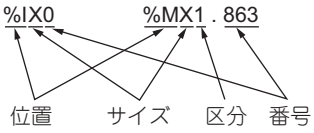
*1: FXCPU では使用できません。

●位置

データを割り当てる位置を，入力／出力／内部の 3 種類に分けて示す大分類です。

デバイス表記に対応する表記の原則は，下記のとおりです。

例）



- X，J¥X（X デバイス）：I（入力）
- Y，J¥Y（Y デバイス）：Q（出力）
- 上記以外のデバイス　：M（内部）

●サイズ

データの大きさを示す分類です。

デバイス表記に対応する表記の原則は下記のとおりです。

- ビットデバイス　　：X（ビット）
- ワードデバイス　　：W（ワード），D（ダブルワード），L（ロングワード）

●区分

上記の位置とサイズだけでは区別できないデバイスの種別を示すための小分類です。

デバイスの X，Y は，区分を扱いません。

デバイス表記に対応する表記は下記を参照してください。

☞ 4.5.3 項　デバイス表記とアドレス表記

ポイント

ロングワードはユニバーサルモデル QCPU/LCPU の倍精度実数演算命令で使います。

4.5.3 デバイス表記とアドレス表記

デバイスとアドレスの表記の仕方と，対応関係を示します。

(1) デバイスとアドレスの対応

デバイスとアドレスの表記方法の対応を下記に示します。

(a) QCPU(Q モード)/LCPU

表 4.5.3-1 デバイスとアドレスの対応（1/2）

デバイス			表記方法		デバイスとアドレスの対応例	
			デバイス	アドレス	デバイス	アドレス
入力		X	Xn	%IXn	X7FF	%IX2047
出力		Y	Yn	%QXn	Y7FF	%QX2047
内部リレー		M	Mn	%MX0.n	M2047	%MX0.2047
ラッチリレー		L	Ln	%MX8.n	L2047	%MX8.2047
アナンシェータ		F	Fn	%MX7.n	F1023	%MX7.1023
特殊リレー		SM	SMn	%MX10.n	SM1023	%MX10.1023
ファンクション入力		FX	FXn	対応なし	FX10	対応なし
ファンクション出力		FY	FYn	対応なし	FY10	対応なし
エッジリレー		V	Vn	%MX9.n	V1023	%MX9.1023
ダイレクトアクセス入力		DX	DXn	%IX1.n	DX7FF	%IX1.2047
ダイレクトアクセス出力		DY	DYn	%QX1.n	DY7FF	%QX1.2047
タイマ	接点	TS	Tn	%MX3.n	TS511	%MX3.511
	コイル	TC	Tn	%MX5.n	TC511	%MX5.511
	現在値	TN	Tn	%MW3.n %MD3.n	TN511 T511	%MW3.511 %MD3.511
カウンタ	接点	CS	Cn	%MX4.n	CS511	%MX4.511
	コイル	CC	Cn	%MX6.n	CC511	%MX6.511
	現在値	CN	Cn	%MW4.n %MD4.n	CN511 C511	%MW4.511 %MD4.511
積算タイマ	接点	STS	STn	%MX13.n	STS511	%MX13.511
	コイル	STC	STn	%MX15.n	STC511	%MX15.511
	現在値	STN	STn	%MW13.n %MD13.n	STN511 ST511	%MW13.511 %MD13.511
データレジスタ		D	Dn	%MW0.n %MD0.n	D11135	%MW0.11135 %MD0.11135
特殊レジスタ		SD	SDn	%MW10.n %MD10.n	SD1023	%MW10.1023 %MD10.1023
ファンクション レジスタ		FD	FDn	対応なし	FD0	対応なし
リンクリレー		B	Bn	%MX1.n	B7FF	%MX1.2047
リンク特殊リレー		SB	SBn	%MX11.n	SB3FF	%MX11.1023
リンクレジスタ		W	Wn	%MW1.n %MD1.n	W7FF	%MW1.2047 %MD1.2047
リンク特殊レジスタ		SW	SWn	%MW11.n %MD11.n	SW3FF	%MW11.1023 %MD11.1023
インテリジェント機能 ユニットデバイス		G	Ux¥Gn	%MW14.x.n %MD14.x.n	U0¥G65535	%MW14.0.65535 %MD14.0.65535
ファイルレジスタ		R	Rn	%MW2.n %MD2.n	R32767	%MW2.32767 %MD2.32767
ポインタ		P	Pn	""（空文字）	P299	対応なし
割込みポインタ		I	In	対応なし	−	−
ネスティング		N	Nn	対応なし	−	−
インデックスレジスタ		Z	Zn	%MW7.n %MD7.n	Z9	%MW7.9 %MD7.9

表 4.5.3-1 デバイスとアドレスの対応（2/2）

デバイス		表記方法		デバイスとアドレスの対応例	
		デバイス	アドレス	デバイス	アドレス
ステップリレー	S	Sn	%MX2.n	S127	%MX2.127
SFC 移行デバイス	TR	TRn	%MX18.n	TR3	%MX18.3
SFC ブロック	BL	BLn	%MX17.n	BL3	%MX17.3
リンク入力	J	Jx¥Xn	%IX16.x.n	J1¥X1FFF	%IX16.1.8191
リンク出力		Jx¥Yn	%QX16.x.n	J1¥Y1FFF	%QX16.1.8191
リンクリレー		Jx¥Bn	%MX16.x.1.n	J2¥B3FFF	%MX16.2.1.16383
リンクレジスタ		Jx¥Wn	%MW16.x.1.n %MD16.x.1.n	J2¥W3FFF	%MW16.2.1.16383 %MD16.2.1.16383
リンク特殊リレー		Jx¥SBn	%MX16.x.11.n	J2¥SB1FF	%MX16.2.11.511
リンク特殊レジスタ		Jx¥SWn	%MW16.x.11.n %MD16.x.11.n	J2¥SW1FF	%MW16.2.11.511
ファイルレジスタ	ZR	ZRn	%MW12.n %MD12.n	ZR32767	%MW12.32767 %MD12.32767

(b) FXCPU

表 4.5.3-2 デバイスとアドレスの対応

デバイス			表記方法		デバイスとアドレスの対応例	
			デバイス	アドレス	デバイス	アドレス
入力		X	Xn	%IXn	X367	%IX247
出力		Y	Yn	%QXn	Y367	%QX247
補助リレー		M	Mn	%MX0.n	M499	%MX0.499
タイマ	接点	TS	Tn	%MX3.n	TS191	%MX3.191
	コイル	TC	Tn	%MX5.n	TC191	%MX5.191
	現在値	TN	Tn	%MW3.n %MD3.n	TN191 T190	%MW3.191 %MD3.190
カウンタ	接点	CS	Cn	%MX4.n	CS99	%MX4.99
	コイル	CC	Cn	%MX6.n	CC99	%MX6.99
	現在値	CN	Cn	%MW4.n %MD4.n	CN99 C98	%MW4.99 %MD4.98
データレジスタ		D	Dn	%MW0.n %MD0.n	D199 D198	%MW0.199 %MD0.198
インテリジェント機能ユニットデバイス		G	Ux¥Gn	%MW14.x.n %MD14.x.n	U0¥G09	%MW14.0.10 %MD14.0.9
拡張レジスタ		R	Rn	%MW2.n %MD2.n	R32767 R32766	%MW2.32767 %MD2.32766
拡張ファイルレジスタ		ER	ERn	対応なし	−	−
ポインタ		P	Pn	""（空文字）	P4095	対応なし
割込みポインタ		I	In	対応なし	−	−
ネスティング		N	Nn	対応なし	−	−
インデックスレジスタ		Z	Zn	%MW7.n %MD7.n	Z7 Z6	%MW7.7 %MD7.6
		V	Vn	%MV6.n	V7	%MW6.7
ステート		S	Sn	%MX2.n	S4095	%MX2.4095

(2) ビットデバイスの桁指定

ビットデバイスの桁指定をする場合の，デバイスとアドレスの対応関係を示します。

表 4.5.3-3 桁指定での表記の対応

デバイス	アドレス
K[桁数][デバイス名][デバイス番号] （桁数：1 ～ 8）	%[メモリ領域の位置][サイズ]19.[桁数].[区分].[番号] （桁数：1 ～ 8）

• 対応例

デバイス	アドレス
K1X0	%IW19.1.0
K4M100	%MW19.4.0.100
K8M100	%MD19.8.0.100
K2Y7E0	%QW19.2.2016

(3) ワードデバイスのビット指定

ワードデバイスのビット指定をする場合の，デバイスとアドレスの対応関係を示します。

表 4.5.3-4 ビット指定での表記の対応

デバイス	アドレス
[デバイス名][デバイス番号].[ビット番号] （ビット番号：0 ～ F）	%[メモリ領域の位置]X[区分].[デバイス番号].[ビット番号]

• 対応例

デバイス	アドレス
D11135.C	%MX0.11135.12
SD1023.F	%MX10.1023.15

ポイント

•インデックス修飾，桁指定，ビット指定について

ラベルへのインデックス修飾，桁指定，ビット指定はできません。

4.6 インデックス修飾

(1) インデックス修飾の概要

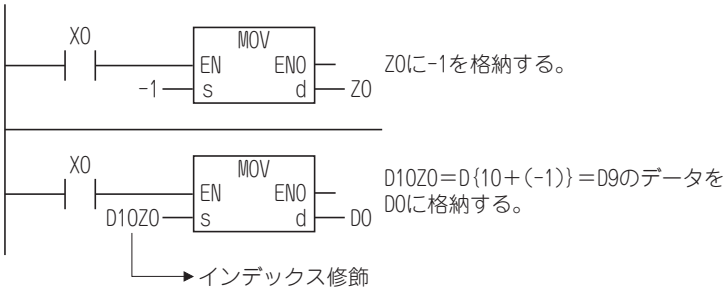
(a) インデックス修飾は，インデックスレジスタを使用した間接設定です。
シーケンスプログラムでインデックス修飾を使用すると，使用するデバイスは，(直接指定しているデバイス番号) + (インデックスレジスタの内容) になります。
たとえば D2Z2 を指定しているとき，Z2 の内容が 3 の場合は，D(2+3)=D5 が対象になります。

(b) ユニバーサルモデル QCPU，LCPU，FXCPU の場合，16 ビットの範囲で修飾するほかに，32 ビットの範囲で修飾することができます。

(2) 16 ビットによるインデックス修飾

(a) 16 ビットの範囲で修飾する場合

各インデックスレジスタには，−32768 ～ 32767 の設定ができます。*1
インデックス修飾は次のようになります。



*1: インデックスレジスタの仕様については，使用している CPU ユニットのユーザーズマニュアル（機能解説･プログラム基礎編）を参照してください。

(b) インデックス修飾可能デバイス（QCPU(Q モード)，LCPU の場合）

インデックス修飾は，下記に示す制約を除き接点，コイル，基本命令，応用命令で使用するデバイスに使用できます。ラベルへのインデックス修飾はできません。

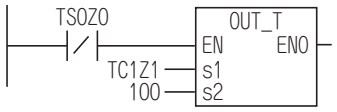
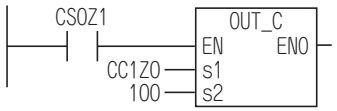
1) インデックス修飾が不可のデバイス

デバイス	内　容
E	浮動小数点データ
$	文字列データ
□.□(D0.1 など)	ワードデバイスのビット指定
FX,FY,FD	ファンクションデバイス
P	ラベルとしてのポインタ
I	ラベルとしての割込みポインタ
Z	インデックスレジスタ
S	ステップリレー *3
TR	SFC 移行デバイス*2
BL	SFC ブロックデバイス*2 *3

*2: SFC 移行デバイス，SFC ブロックデバイスは，SFC 用のデバイスです。
使用方法については，下記マニュアルを参照してください。
• MELSEC-Q/L/QnA プログラミングマニュアル（SFC 編）

*3: ユニバーサルモデル高速タイプ QCPU およびユニバーサルモデルプロセス CPU の SFC ブロック (BL) とステップリレー (S) は，下記の範囲でインデックス修飾できます。
• SFC ブロック (BL) は BL0 ～ BL319 の範囲
• ステップリレー (S) はパラメータのデバイス設定で設定された範囲
ただし，SFC ブロック内のステップリレー (S) を指定する場合は，S0 ～ S511 の範囲でインデックス修飾できます。

2) 使用インデックスレジスタに制約のあるデバイス

デバイス	内　容	使用例
T	・タイマの接点，コイルには， Z0,Z1 のみ使用可	
C	・カウンタの接点，コイルには Z0,Z1 のみ使用可	

(c) インデックス修飾可能デバイス（FXCPU の場合）
インデックス修飾可能なデバイスは下記です。

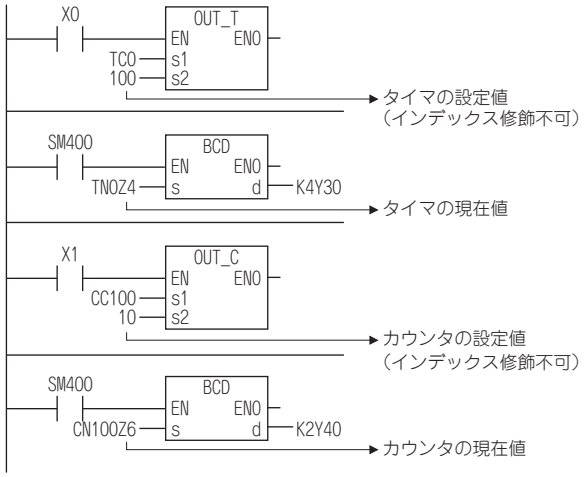
デバイス	内　容
M,S,T,C,D,R,KnM,KnS,P,K	10 進数デバイス・数値
X,Y,KnX,KnY	8 進数デバイス
H	16 進数数値

1) 使用インデックスレジスタに制約のあるデバイス
FXCPU の場合，下記の内容にご注意ください。

- 基本命令で使用するデバイスへのインデックス修飾は FX3U，FX3UC のみ可能です。
- 32 ビットカウンタ，特殊補助リレーへのインデックス修飾はできません。

備 考

タイマ，カウンタの現在値には，インデックスレジスタ番号の使用制限はありません。



(d) インデックス修飾をした場合と実際の処理デバイスは次のようになります。
(Z0=20,Z1=−5 のとき)

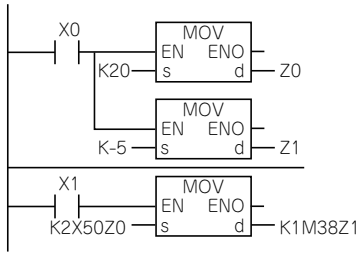
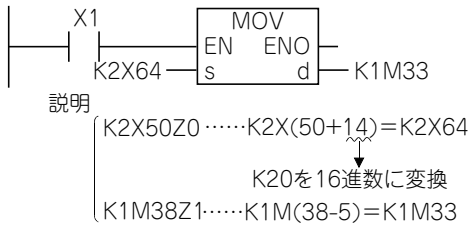
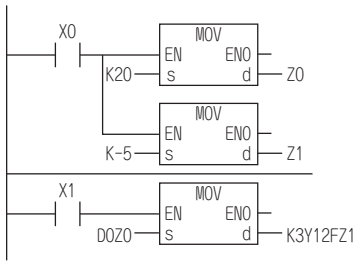
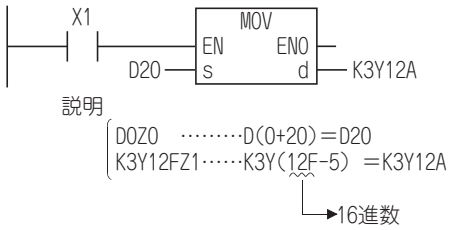
回路例	実際の処理デバイス
	
	

図 4.6-1 回路例と実際の処理デバイス

(3) 32 ビットによるインデックス修飾

(ユニバーサルモデル QCPU（Q00UJCPU を除く），LCPU，FXCPU の場合)

ユニバーサルモデル QCPU（Q00UJCPU を除く），LCPU の場合，32 ビットによるインデックス修飾する場合でのインデックスレジスタ指定方法を，下記の 2 つから選択することができます。

- 32 ビットインデックス修飾に使用するインデックスレジスタの範囲を指定する。
- “ZZ” 表現による 32 ビットインデックス修飾を指定する。

FXCPU の場合，32 ビットによるインデックス修飾はインデックスレジスタ V(V0 ～) と Z(Z0 ～) の組合せで修飾します。

ポイント

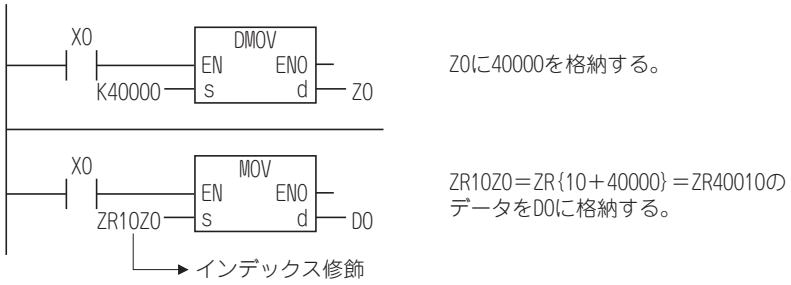
“ZZ” 表現による 32 ビットインデックス修飾は，下記 CPU ユニットのみ使用可能です。

- シリアル No. の上 5 桁が “10042” 以降の QnU(D)(H)CPU（Q00UJCPU を除く）
- QnUDE(H)CPU
- QnUDVCPU
- QnUDPVCPU
- LCPU

(a) 32 ビットインデックス修飾に使用するインデックスレジスタの範囲を指定する場合

1) 各インデックスレジスタには，−2147483648 ～ 2147483647 の設定ができます。

インデックス修飾は次のようになります。



2) 指定方法

32 ビットの範囲で修飾する場合は，PC パラメータの《デバイス設定》⇒“ZR デバイスのインデックス修飾設定” で，使用するインデックスレジスタの先頭番号を指定します。

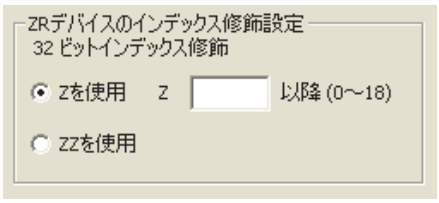


図 4.6-2 ZR デバイスのインデックス修飾設定パラメータの設定画面

ポイント

PC パラメータのデバイス設定で，使用するインデックスレジスタの先頭番号を変更した場合，パラメータのみを変更または PC 書込しないでください。必ずプログラムと一緒に PC 書込を行ってください。
強制的に書込みを行うと，CAN'T EXE. PRG.(エラーコード：2500) になります。

3) インデックス修飾可能デバイス

インデックス修飾は，下記に示すデバイスのみ使用できます。

デバイス	内　容
ZR	連番アクセス方式ファイルレジスタ
D	拡張データレジスタ
W	拡張リンクレジスタ
M *3	内部リレー
B *3	リンクリレー
D *3	データレジスタ
W *3	リンクレジスタ
Un¥B *3	リンクリレー
Un¥W *3	リンクレジスタ

*3　ユニバーサルモデル高速タイプ QCPU およびユニバーサルモデルプロセス CPU のみ使用できます。

4) インデックスレジスタの使用範囲

32 ビットの範囲で修飾する場合でのインデックスレジスタの使用範囲を下表に示します。

32 ビットの範囲のインデックス修飾では，指定したインデックスレジスタ（Zn）と，連続した次のインデックスレジスタ（Zn+1）を使用するため，使用しているインデックスレジスタが重ならないように注意してください。

設定値	使用するインデックスレジスタ	設定値	使用するインデックスレジスタ
Z0	Z0，Z1	Z10	Z10，Z11
Z1	Z1，Z2	Z11	Z11，Z12
Z2	Z2，Z3	Z12	Z12，Z13
Z3	Z3，Z4	Z13	Z13，Z14
Z4	Z4，Z5	Z14	Z14，Z15
Z5	Z5，Z6	Z15	Z15，Z16
Z6	Z6，Z7	Z16	Z16，Z17
Z7	Z7，Z8	Z17	Z17，Z18
Z8	Z8，Z9	Z18	Z18，Z19
Z9	Z9，Z10	Z19	使用不可

5) インデックス修飾をした場合と実際の処理デバイスは次のようになります。

（Z0（32 ビット）=100000, Z2（32 ビット）=-20 のとき ）

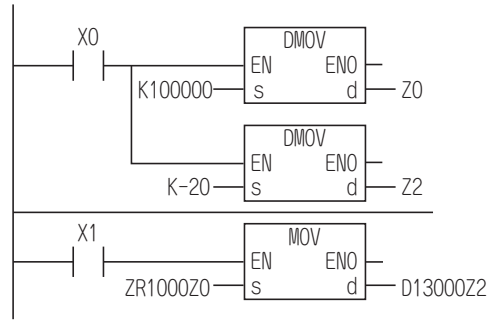
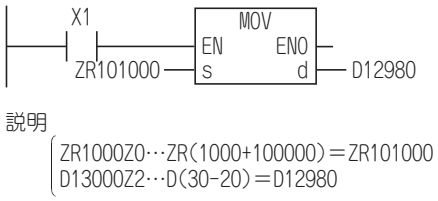
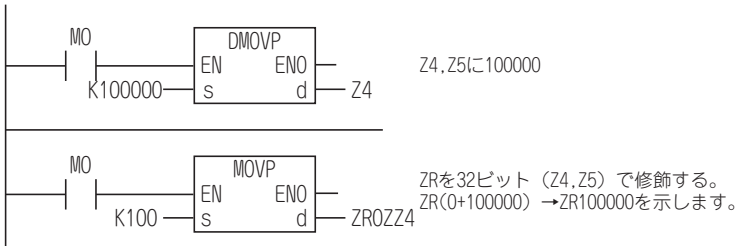
回路例	実際の処理デバイス
	

図 4.6-3 回路例と実際の処理デバイス

(b) “ZZ” 表現による 32 ビットインデックス修飾を指定する場合

1) “ZR0ZZ4” のように “ZZ" 表現によるインデックス修飾を指定することで，任意のインデックスレジスタにより 32 ビットインデックス修飾を指定できます。

“ZZ” 表現による 32 ビットインデックス修飾は次のようになります。



2) 指定方法

“ZZ” 表現による 32 ビットインデックス修飾する場合は，PC パラメータの《デバイス設定》⇒ “ZR デバイスのインデックス修飾設定 ” で “ZZ を使用 ” を設定します。

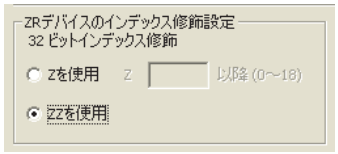


図 4.6-4 ZR デバイスのインデックス修飾設定パラメータの設定画面

3) インデックス修飾可能デバイス

インデックス修飾は，下記に示すデバイスのみ使用できます。

デバイス	内　容
ZR	連番アクセス方式ファイルレジスタ
D	拡張データレジスタ
W	拡張リンクレジスタ

4) インデックスレジスタの使用範囲

“ZZ” 表現による 32 ビットインデックス修飾する場合でのインデックスレジスタの使用範囲を下表に示します。

“ZZ” 表現による 32 ビットインデックス修飾を指定するときは，ZRmZZn の形式で指定します。

ZRmZZn を指定することにより，ZRm のデバイス番号を Zn，Zn+1 の 32 ビット値で修飾します。

“ZZ” 表現 *2	使用するインデックスレジスタ	“ZZ” 表現 *4	使用するインデックスレジスタ
□ZZ0	Z0，Z1	□ZZ10	Z10，Z11
□ZZ1	Z1，Z2	□ZZ11	Z11，Z12
□ZZ2	Z2，Z3	□ZZ12	Z12，Z13
□ZZ3	Z3，Z4	□ZZ13	Z13，Z14
□ZZ4	Z4，Z5	□ZZ14	Z14，Z15
□ZZ5	Z5，Z6	□ZZ15	Z15，Z16
□ZZ6	Z6，Z7	□ZZ16	Z16，Z17
□ZZ7	Z7，Z8	□ZZ17	Z17，Z18
□ZZ8	Z8，Z9	□ZZ18	Z18，Z19
□ZZ9	Z9，Z10	□ZZ19	使用不可

*4　□は修飾対象のデバイス名（ZR, D, W）を示します。

5）“ZZ” 表現による 32 ビットインデックス修飾を指定した場合と実際の処理デバイスは次のようになります。

(Z0(32 ビット)=100000, Z2(32 ビット)=-20 のとき)

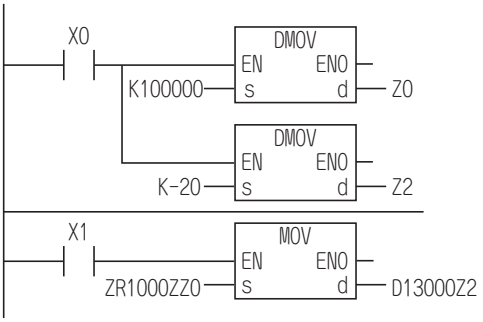
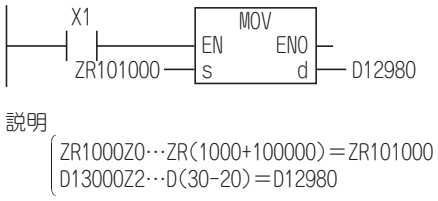
回路例	実際の処理デバイス
	

図 4.6-5 回路例と実際の処理デバイス

6）“ZZ” 表現を使用できる機能

“ZZ” 表現による 32 ビットインデックス修飾の指定は，下記に示す機能で使用することができます。

No.	機能名称・説明
1	プログラム中の命令でのデバイス指定
2	デバイス登録モニタ
3	デバイステスト
4	実行条件付きデバイステスト
5	モニタ条件設定
6	サンプリングトレース (トレースポイント (デバイス指定)，トレース対象デバイス)
7	データロギング機能 (収集間隔 (デバイス指定)，ロギング対象データ)

ポイント

“DMOV K100000 ZZ0” のように，ZZn を単独でデバイスのように扱うことはできません。“ZZ" 表現による 32 ビットインデックス修飾を指定するためにインデックスレジスタへ値を設定するときには，Zn(Z0 ～ Z19) に対して設定してください。

各機能には，ZZn を単独で入力することができません。

(c) FXCPU での 32 ビットによるインデックス修飾

インデックスレジスタ V(V0 ～) と Z(Z0 ～) を組み合わせて，32 ビットで修飾します。

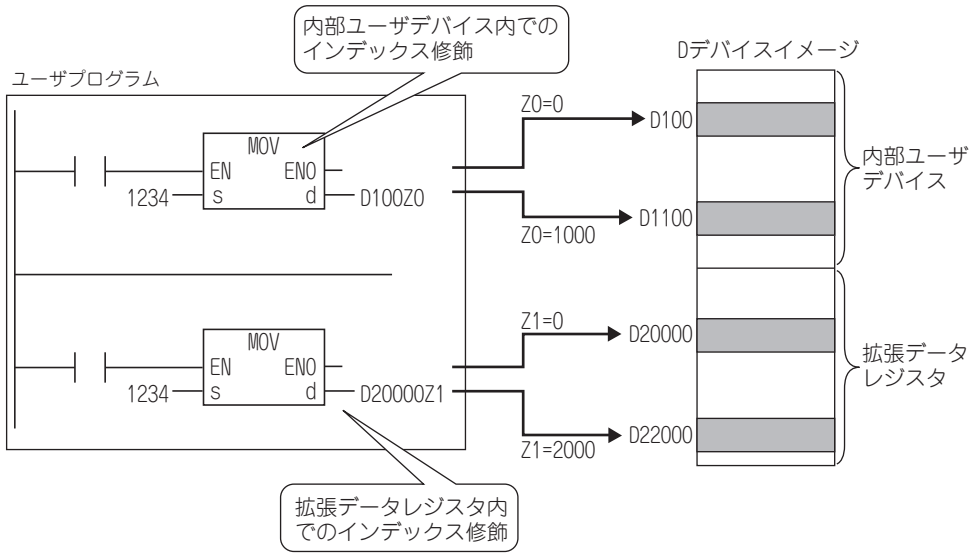
V 側が上位，Z 側が下位となります。下位側の Z を指定すると，これと組となる V 側を含め 32 ビットレジスタとして働きます。

上位側である V を指定しても修飾は行なわれません。

例：Z4 を指定すると，V4，Z4 で 32 ビットレジスタとして修飾します。

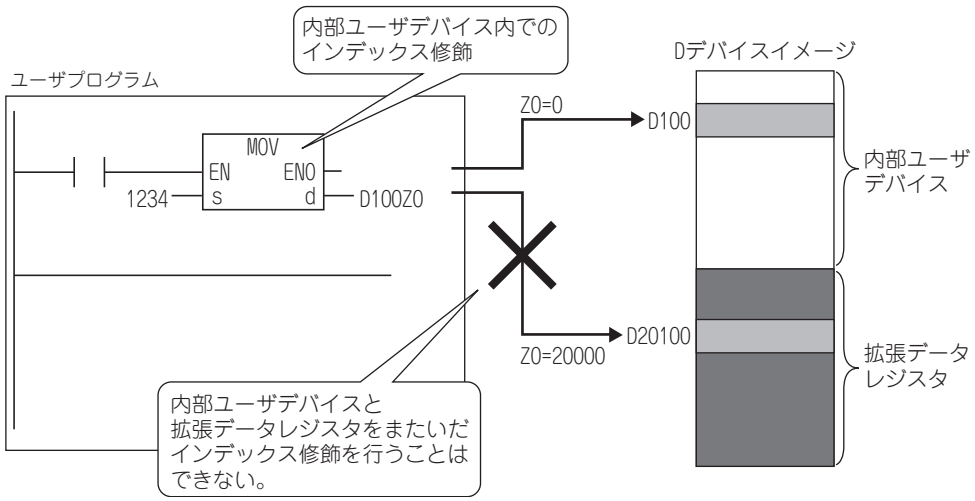
設定値	使用するインデックスレジスタ
Z0	V0，Z0
Z1	V1，Z1
Z2	V2，Z2
Z3	V3，Z3
Z4	V4，Z4
Z5	V5，Z5
Z6	V6，Z6
Z7	V7，Z7

(4) 拡張データレジスタ（D），拡張リンクレジスタ（W）でインデックス修飾する場合
(ユニバーサルモデル QCPU（Q00UJCPU を除く），LCPU)
内部ユーザデバイスのデータレジスタ (D)，リンクレジスタ (W) でのインデックス修飾と同様に，拡張データレジスタ (D)，拡張リンクレジスタ (W) の範囲内でインデックス修飾によるデバイス指定を使用することができます。



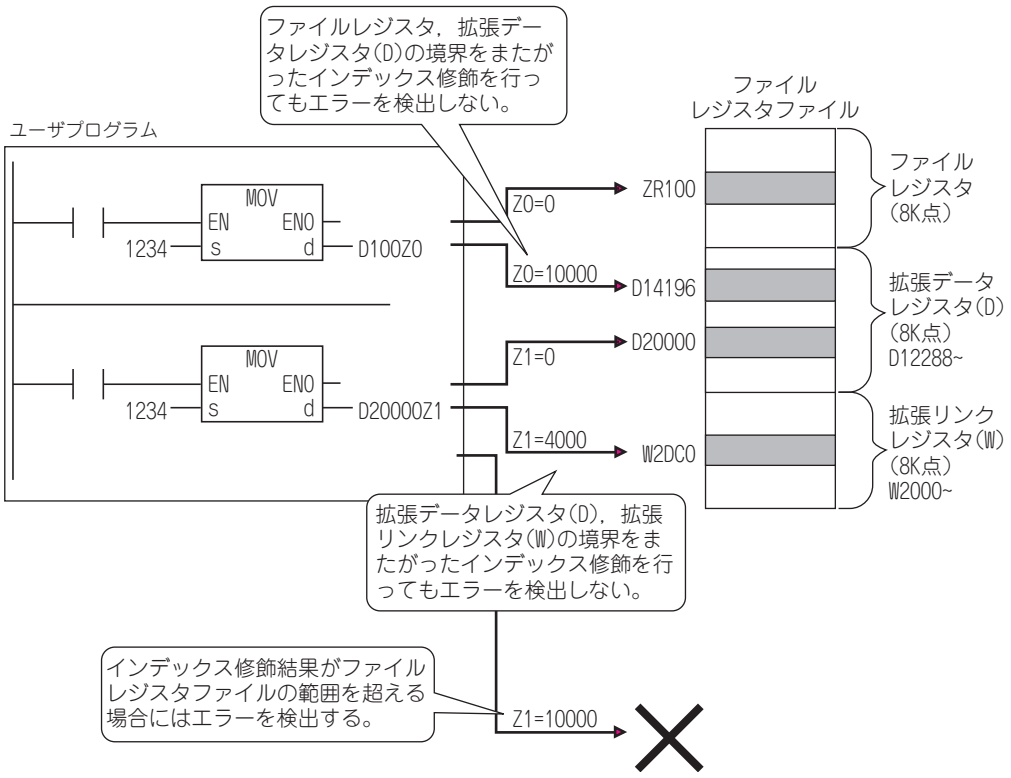
1) 内部ユーザデバイスと拡張データレジスタ（D），拡張リンクレジスタ（W）をまたぐインデックス修飾

内部ユーザデバイスと拡張データレジスタ（D），拡張リンクレジスタ（W）をまたいでインデックス修飾の指定はできません。インデックス修飾時のデバイス範囲チェックが有効になっている場合に，エラーになります。
（エラーコード：4101）



2) ファイルレジスタ (ZR)，拡張データレジスタ (D)，拡張リンクレジスタ (W) をまたぐインデックス修飾

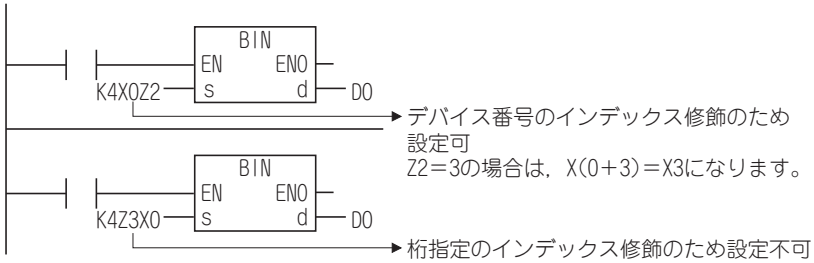
ファイルレジスタ (ZR)，拡張データレジスタ (D)，拡張リンクレジスタ (W) をまたぐインデックス修飾を行っても，エラーになりません。
ただし，ファイルレジスタ (ZR)，拡張データレジスタ (D)，拡張リンクレジスタ (W) のインデックス修飾結果がファイルレジスタファイルの範囲を超える場合は，エラーになります。(エラーコード：4101)



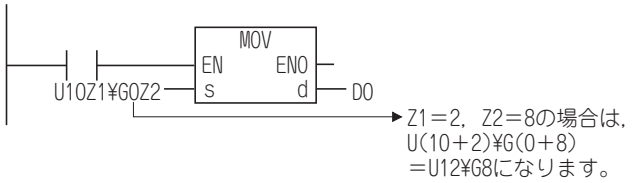
(5) その他のインデックス修飾をする場合

(a) ビットデータ

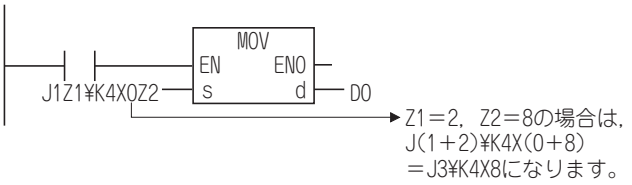
桁指定を行っている場合，デバイス番号はインデックス修飾できます。
ただし桁指定は，インデックス修飾できません。



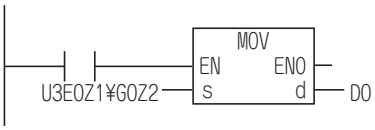
(b) インテリジェント機能ユニットデバイス *4 では，インテリジェント機能ユニットの先頭入出力番号とバッファメモリアドレスを共にインデックス修飾できます。



(c) リンクダイレクトデバイス *5 では，ネットワーク No. とデバイス番号を共にインデックス修飾できます。



(d) マルチ CPU 間共有デバイス *6 では，CPU ユニットの先頭入出力番号と CPU 共有メモリアドレスを共にインデックス修飾できます。



＊5：インテリジェント機能ユニットデバイス，リンクダイレクトデバイスは，使用している CPU ユニットのユーザーズマニュアル（機能解説・プログラム基礎編）を参照してください。
＊6：マルチ CPU 間共有デバイスは，使用している CPU ユニットのユーザーズマニュアル（機能解説・プログラム基礎編）を参照してください。

(e) 拡張データレジスタ (D)，拡張リンクレジスタ (W) の 32 ビットによるインデックス修飾する場合
（ユニバーサルモデル QCPU（Q00UJCPU を除く），LCPU の場合）
拡張データレジスタ (D)，拡張リンクレジスタ (W) のインデックス修飾を行う場合，ファイルレジスタ (ZR) のインデックス修飾と同様に 32 ビットの範囲でインデックス修飾を下記 2 つの方法で行うことができます。

- 32 ビットインデックス修飾に使用するインデックスレジスタの範囲を指定する。
- “ZZ” 表現による 32 ビットインデックス修飾を指定する。

ポイント

“ZZ” 表現による 32 ビットインデックス修飾は，下記 CPU ユニットのみ使用可能です。

- シリアル No. の上 5 桁が “10042” 以降の QnU(D)(H)CPU（Q00UJCPU を除く）
- QnUDE(H)CPU
- QnUDVCPU
- QnUDPVCPU
- LCPU

(6) 注意事項

(a) 命令／応用関数／ファンクション／ファンクションブロックの引数にインデックス修飾を使用する場合

PC パラメータの《デバイス設定》⇒ “ZR デバイスのインデックス修飾設定 ” で “ZZ を使用 ” にチェック時，命令／応用関数／ファンクション／ファンクションブロックの引数に Z デバイスを使用すると，コンパイル後に "ZZ" 表現に変換されます。そのため，意図しないデバイスにアクセスする場合があります。

“ZZ を使用 ” にチェックした場合は，命令／応用関数／ファンクション／ファンクションブロックの引数には ZZ デバイスを使用してください。

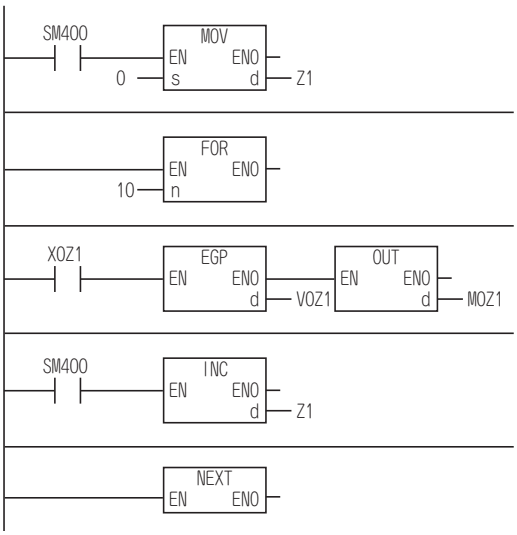
(b) FOR ～ NEXT 命令間でインデックス修飾する場合

FOR ～ NEXT 命令間ではエッジリレー（V）を使用することで，パルス出力ができます。

ただし，PLS/PLF/ パルス化（□P）命令でのパルス出力はできません。

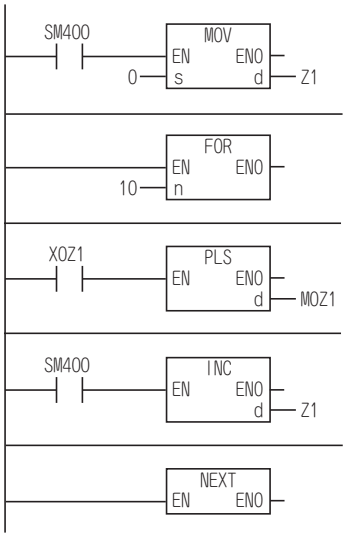
［エッジリレーを使用する場合］

（M0Z1 が正常にパルス出力される。）



［エッジリレーを使用しない場合］

（M0Z1 が正常にパルス出力されない。）



備 考

X0Z1 の ON/OFF 情報を，エッジリレーの V0Z1 で記憶します。
たとえば X0 の ON/OFF 情報を V0 で記憶し，X1 の ON/OFF 情報を V1 で記憶していきます。

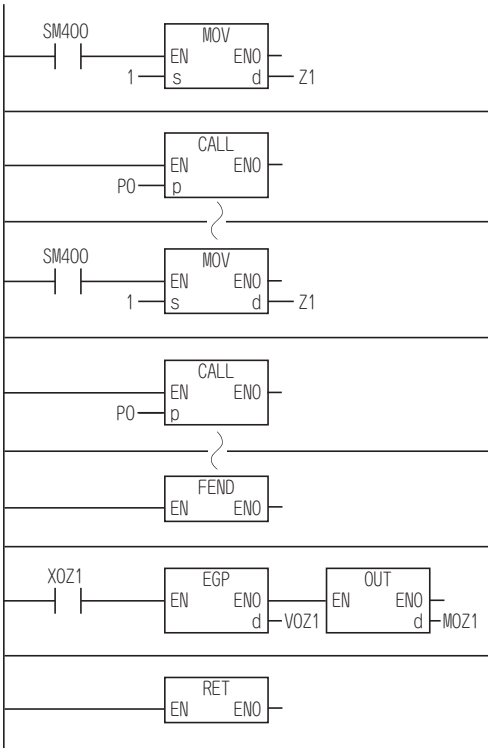
ポイント

FOR ～ NEXT 命令間で配列の添え字にラベルを使用する場合，Z0，Z1 は使用しないでください。

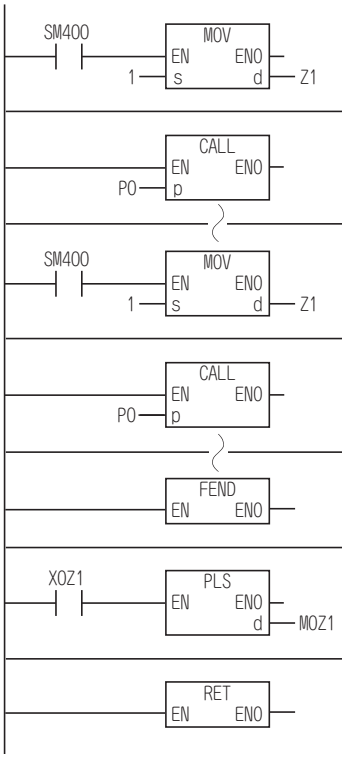
(c) CALL 命令でインデックス修飾する場合

CALL 命令ではエッジリレー（V）を使用することで，パルス出力ができます。
ただし，PLS/PLF/ パルス化（□P）命令でのパルス出力はできません。

［エッジリレーを使用する場合］
（M0Z1 が正常にパルス出力される。）



［エッジリレーを使用しない場合］
（M0Z1 が正常にパルス出力されない。）



(d) インデックス修飾時のデバイス範囲チェック

1) ベーシックモデル QCPU，ハイパフォーマンスモデル QCPU，プロセス CPU，二重化 CPU，FXCPU の場合

インデックス修飾時はデバイス範囲チェックを行いません。
ベーシックモデル QCPU，ハイパフォーマンスモデル QCPU，プロセス CPU，二重化 CPU 時はインデックス修飾を行った結果，ユーザにて指定したデバイスの範囲を超えた場合は，エラーにならずに他のデバイスにデータが書き込まれます。（ただし，インデックス修飾を行った結果，ユーザ用のデバイス範囲を超えてシステム用のデバイスに書き込んだ場合は，エラーになります。
（エラーコード：1103））
FXCPU 時は，演算エラーとなります。（エラーコード：6706）
インデックス修飾を使用したプログラムの作成には，十分注意してください。

2) ユニバーサルモデル QCPU，LCPU の場合

インデックス修飾時は，デバイス範囲チェックを行います。
また，PC パラメータによりデバイス範囲チェックを行わないようにすることもできます。

(e) 16 ビット⇔ 32 ビットインデックス修飾範囲の変更

16 ビット⇔ 32 ビットインデックス修飾範囲の変更をする場合は，プログラム内のインデックス修飾箇所を見直してください。
32 ビットの範囲のインデックス修飾では，指定したインデックスレジスタ（Zn）と，連続した次のインデックスレジスタ（Zn+1）を使用するため，使用しているインデックスレジスタが重ならないように注意してください。

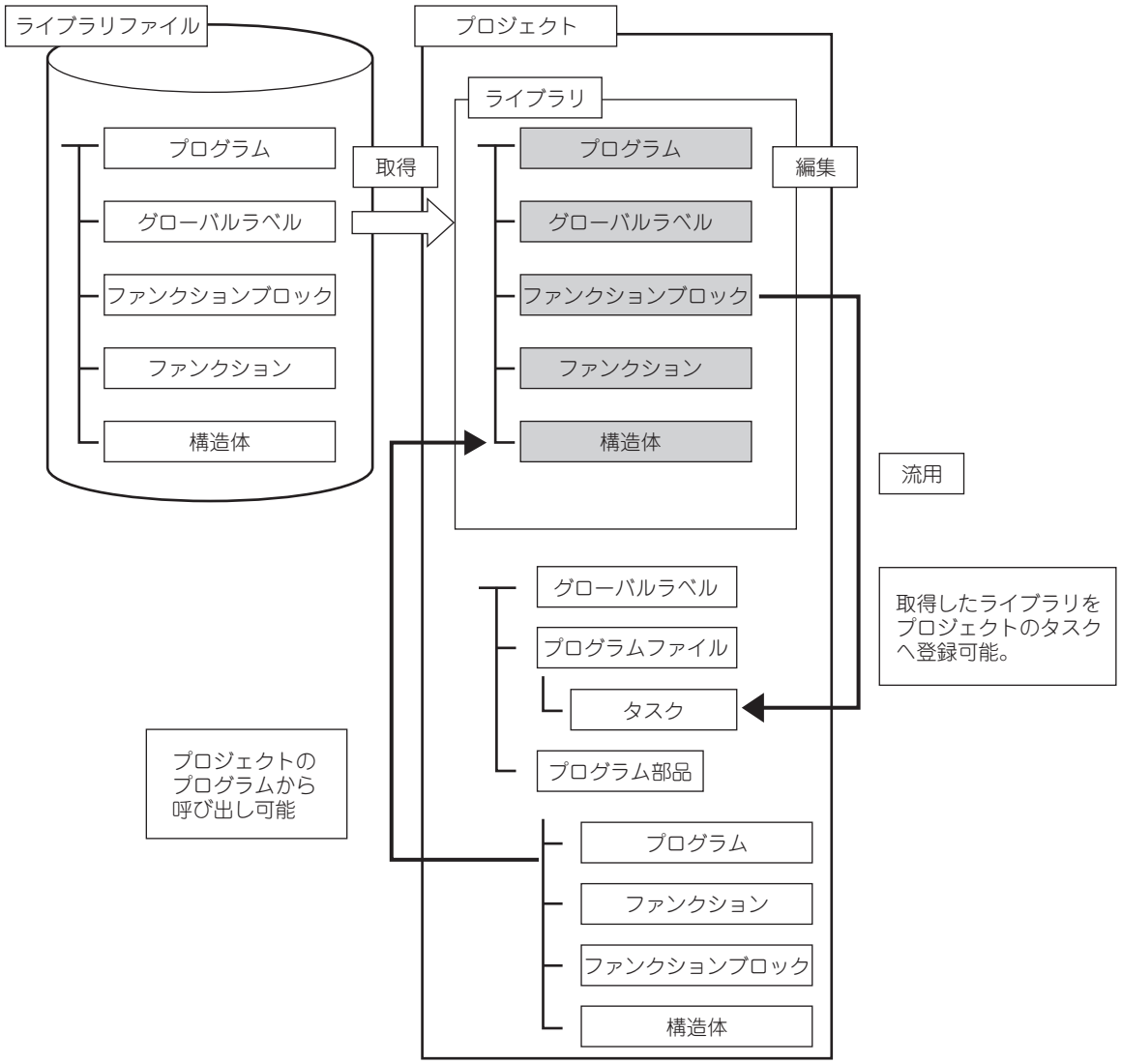
4.7 ライブラリ

ライブラリとは，プログラム部品やグローバルラベル，構造体などを 1 つのファイルにまとめ，各プロジェクトで共通して使用することを目的にしたデータの集まりです。

ライブラリを使用すると，下記のような利点があります。

- 各プロジェクトでライブラリを取得し，ライブラリ内のデータを共有して利用することができます。
- ライブラリを，部品の機能ごとに分けて作成できるので，再利用したい部品の確認が容易になります。
- ライブラリに登録された部品に変更を加えた場合，その部品を使用しているプロジェクトへも反映されます

下記に，ライブラリの部品をプロジェクトで使用するときのデータの流れを示します。



4.7.1 ユーザライブラリ

ユーザライブラリは，ユーザが作成した共通使用可能な構造体，グローバルラベル，プログラム部品などをまとめて保存するライブラリです。

(1) ユーザライブラリの構成

下記のものが，ユーザライブラリに登録できます。

表 4.7.1-1 ユーザライブラリの構成

構成名	内容
構造体	ライブラリ内のプログラム部品フォルダで使用している構造体の定義や，プロジェクト内のプログラムで使用する構造体の定義を格納します。
グローバルラベル	ライブラリ内のプログラム部品フォルダで使用しているグローバルラベルの定義を格納します。
プログラム部品	ライブラリとして使用可能なプログラム，ファンクション，ファンクションブロックを格納します。

4.8 名称を付けるときの注意事項

ラベル，ファンクションブロックのインスタンス，構造体ラベルの名称を付けるときの条件を示します。

●条件

(1) 32 字以内の文字列で指定してください。

(2) 予約語を使用しないでください。
予約語は下記を参照してください。
☞ 付録 2 ラベル名やデータ名で使用できない文字列

(3) 下記文字を使用してください。
半角英数字，半角カタカナ，アンダスコア (_)，全角文字

(4) アンダスコアを最後に使用しないでください。
また，アンダスコアは 2 つ以上連続して使用しないでください。

(5) スペースを使用しないでください。

(6) 先頭文字に半角の数字を使用しないでください。

(7) 定数は使用できません。
(“H” あるいは “h” で始まる識別子の場合，“H” あるいは “h” の直後に 16 進数 (0 ～ F) が連続して続く場合（“H” あるいは “h” を含めて最大 9 桁（“H” あるいは “h” の直後に続く 0 は除く））も定数として扱われます。(例：“hab0”))

(8) 基本データ型名は使用できません。

(9) 関数／ FB の部品名は使用できません。

5 プログラムの記述方法

5.1 ST 5-2
5.2 構造化ラダー /FBD 5-13

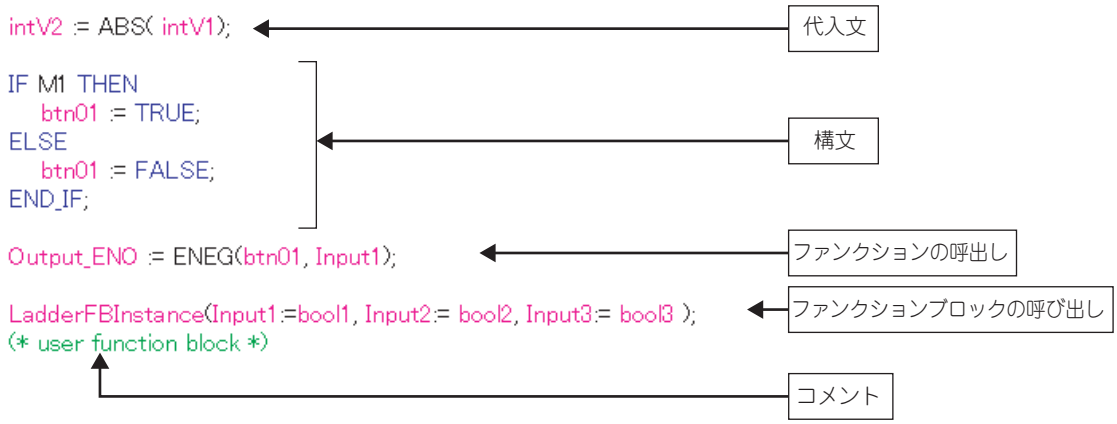
5.1 ST

C 言語などに似た文法構造を持つ，テキスト形式のプログラム言語です。

構文による条件判断や反復などの制御が記述できます。

グラフィック言語（構造化ラダー /FBD）では表現しにくい複雑な処理をプログラミングする場合に適しています。

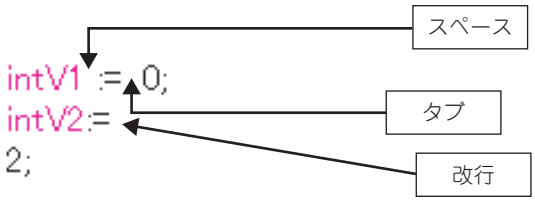
5.1.1 基本書式



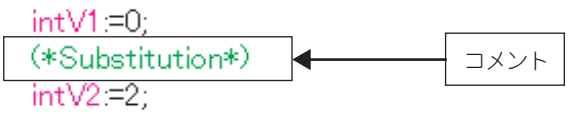
ST でのプログラミングは，演算子と文で構成されます。

文の最後は，必ず “；” を付けます。

スペース，タブ，改行などを，キーワードと識別子の間ならどこにでも挿入できます。



プログラム中にコメントを挿入することができます。コメントの前に “（*”，コメントの後に “*）” を記述します。



コメントの中にコメントを入れると，下記のコンパイルエラーとなります。

コンパイルエラー内容：「構文が解析できません。」，エラーコード：C1200

(* Flag_A = TRUE 制御開始 *) Flag_A = FALSE 制御停止 *)

(* START (* 処理中断 *)再開 終了 *)

5.1.2 ST の演算子

下記に，ST プログラムで使用する演算子と演算子の優先順位を示します。

表 5.1.2-1 ST の演算子

演算子	内容	例	優先順位
( )	丸括弧式	(1+2)*(3+4)	最上位
関数 ( )	関数（パラメータリスト）	ADD_E(bo01, in01, in02, in03)	
**	べき乗	re01:= 2.0 ** 4.4	
NOT	ビットを反転した値	NOT bo01	
* / MOD	乗算 除算 剰余	3 * 4 12 / 3 13 MOD 3	
+ -	加算 減算	in01 + in02 in01 - in02	
<, >, <=, =>	比較	in01 < in02	
= <>	等式 不等式	in01 = in02 in01 <> in02	
AND, &	論理積	bo01 & bo02	
XOR	排他的論理和	bo01 XOR bo02	
OR	論理和	bo01 OR bo02	最下位

1 つの文中に，優先順位が同じ演算子が複数ある場合は，左側の演算子から演算されます。

下記に，演算子と対象データ型，演算結果のデータ型を示します。

表 5.1.2-2 演算子で使用するデータ型

演算子	対象データ型	演算結果型
*, /, +, -	ANY_NUM	ANY_NUM
<, >, <=, >=, =, <>	ANY_SIMPLE	ビット
MOD	ANY_INT	ANY_INT
AND, &, XOR, OR, NOT	ANY_BIT	ANY_BIT
**	ANY_REAL（底） ANY_NUM（指数）	ANY_REAL

5.1.3 ST の構文

下記に，ST プログラムで使用できる構文を示します。

表 5.1.3-1 ST の構文

構文の種類	内容
代入文	代入文
条件文	IF　THEN 条件文，IF　ELSE 条件文，IF　ELSIF 条件文
	CASE 条件文
反復文	FOR　DO 構文
	WHILE　DO 構文
	REPEAT　UNTIL 構文
その他の制御構文	RETURN 構文
	EXIT 構文

（1）代入文

（a）書式

＜左辺＞　:=　＜右辺＞;

（b）説明

代入文は右辺の式の結果を左辺のラベルやデバイスに代入する機能を持ちます。
代入文では，右辺の式の結果と左辺のデータ型が同じである必要があります。

（c）記述例

```
intV1:=0;
intV2:=2;
```

ポイント

代入文では，配列型ラベルや構造体ラベルを使用することもできます。
ただし，代入文の左辺と右辺のデータ型に注意してください。

- 配列型ラベル
  データ型と要素数が左辺と右辺で同じである必要があります。
  また配列型ラベルを使用する場合，要素は指定しないでください。
  ＜記述例＞
  intAry1 := intAry2;

- 構造体ラベル
  データ型（構造体のデータ型）が左辺と右辺で同じである必要があります。
  ＜記述例＞
  dutVar1 := dutVar2;

(2) IF　THEN　条件文

(a) 書式

IF　＜ブール式＞　THEN
＜文・・・＞;
END_IF;

(b) 説明

ブール式（条件式）が真（TRUE）のときに，文を実行します。ブール式が偽（FALSE）の場合，文は実行されません。
ブール式は，単一のビット型変数の状態，あるいは数多くの変数を含んだ複雑な式のブール演算の結果として，真（TRUE）か偽（FALSE）を返す式であればどのような式でも使用できます。

(c) 記述例

```
IF bool1 THEN
    intV1:=intV1 +1;
END_IF;
```

(3) IF　...ELSE　条件文

(a) 書式

IF　＜ブール式＞　THEN
＜文 1・・・＞;
ELSE
＜文 2・・・＞;
END_IF;

(b) 説明

ブール式（条件式）が真（TRUE）のときに，文 1 を実行します。
ブール式の値が偽（FALSE）の場合は文 2 を実行します。

(c) 記述例

```
IF bool1 THEN
    intV3 :=intV3 +1;
ELSE
    intV4 :=intV4 +1;
END_IF;
```

(4) IF　...ELSIF　条件文

(a) 書式

IF　＜ブール式 1 ＞　THEN
＜文 1・・・＞;
ELSIF　＜ブール式 2 ＞　THEN
＜文 2・・・＞;
ELSIF　＜ブール式 3 ＞　THEN
＜文 3・・・＞;
END_IF;

(b) 説明

ブール式（条件式）1 が真 (TRUE) のときに，文 1 を実行します。ブール式 1 の値が偽 (FALSE) でブール式 2 の値が真 (TRUE) の場合は文 2 を実行します。
ブール式 1，2 の値が偽 (FALSE) でブール式 3 の値が真 (TRUE) の場合は文 3 を実行します。

(c) 記述例

```
IF bool1 THEN
    intV1 :=intV1 +1;
ELSIF bool2 THEN
    intV2 :=intV2 +2;
ELSIF bool3 THEN
    intV3 :=intV3 +3;
END_IF;
```

(5) CASE　条件文

(a) 書式

CASE　＜整数式＞　OF
＜整数選択値 1 ＞　：　＜文 1・・・＞;
＜整数選択値 2 ＞　：　＜文 2・・・＞;
.
.
.
＜整数選択値 n ＞　：　＜文 n・・・＞;
ELSE
＜文 n+1・・・＞;
END_CASE;

(b) 説明

CASE 条件文の式の結果は整数値で返されます。この条件文は，例えば単一の整数値や複雑な式の結果の整数値によって，選択文を実行する場合に使用できます。
整数式の値と一致する整数の選択値を持った文が実行され，一致するものがない場合は，ELSE 文に続く文を実行します。

(c) 記述例

```
CASE intV1 OF
    1:bool1 :=TRUE;
    2:bool2 :=TRUE;
ELSE
    intV1 :=intV1 +1;
END_CASE;
```

(6) FOR...DO　構文

(a) 書式

FOR　＜反復変数初期化＞
TO　＜最終値＞
BY　＜増加式＞　DO
＜文・・・＞;
END_FOR;

(b) 説明

FOR...DO 構文は，反復変数の値に応じて，いくつかの文を繰り返し実行します。

(c) 記述例

```
FOR intV1 := 0
    TO 30
    BY 1 DO
    intV3 :=intV1 +1;
END_FOR;
```

(7) WHILE...DO　構文

(a) 書式

WHILE　＜ブール式＞　DO
＜文・・・＞;
END_WHILE;

(b) 説明

WHILE...DO 構文はブール式（条件式）が真 (TRUE) の間，1 つ以上の文を実行します。
ブール式は，文の実行に先立って判定され，ブール式が偽 (FALSE) の場合は，WHILE...DO 内の文が実行されません。WHILE 構文中の＜ブール式＞は，結果が真か偽かを返すものであれば良いため，IF 条件文中の＜ブール式＞で指定可能な式はすべて使用することができます。

(c) 記述例

```
WHILE intV1 = 30 DO
    intV1 :=intV1 +1;
END_WHILE;
```

(8) REPEAT...UNTIL　構文

(a) 書式

REPEAT
＜文・・・＞
UNTIL　＜ブール式＞
END_REPEAT;

(b) 説明

REPEAT...UNTIL 構文は，ブール式（条件式）が偽 (FALSE) の間，1 つ以上の文を実行します。
ブール式は，文の実行後に判定され，値が真 (TRUE) の場合は，REPEAT...UNTIL 内の文が実行されません。
REPEAT 構文中の＜ブール式＞は，結果が真か偽かを返すものであれば良いため，IF 条件文中の＜ブール式＞で指定可能な式はすべて使用することができます。

(c) 記述例

```
REPEAT
    intV1 :=intV1 +1;
    UNTIL intV1 = 30
END_REPEAT;
```

(9) RETURN　構文

(a) 書式

RETURN;

(b) 説明

RETURN 構文はプログラムを途中で終了するために使用します。
RETURN 構文をプログラムで使用すると，RETURN 構文以降の処理がすべて無視され，RETURN が実行される場所からプログラムの最終行までジャンプを行います。

(c) 記述例

```
IF bool1 THEN
   RETURN;
END_IF;
```

(10)EXIT　構文

(a) 書式

EXIT;

(b) 説明

EXIT 構文は反復文の中でのみ使用可能な構文で，反復文を途中で終了させます。
反復ループを実行中に EXIT 構文に達すると，EXIT 構文以降の反復ループ処理は実行されません。反復文を終えた次の行からプログラムの実行を継続します。

(c) 記述例

```
FOR intV1 := 0
    TO 10
    BY 1 DO
    IF intV1 > 10 THEN
        EXIT;
    END_IF;
END_FOR;
```

5.1.4 ST でのファンクションの呼び出し

ST でのファンクションの呼び出しは，下記の記述で行います。

```
ファンクション名 ( 変数 1, 変数 2, ...);
```

ファンクション名に続いて，“()” で引数を囲います。
変数が複数ある場合は “,” で区切ります。

変数へ代入することにより，ファンクションを実行した結果が格納されます。

1) 入力変数が 1 つのファンクション（例：ABS）の場合

```
Output1 := ABS(Input1);
```

2) 入力変数が 3 つのファンクション（例：MAX）の場合

```
Output1 := MAX(Input1, Input2, Input3);
```

3) EN/ENO のあるファンクション（例：MOV）の場合

```
boolENO := MOV(boolEN, Input1, Output1);
```

EN/ENO のあるファンクションの場合，ファンクションを実行した結果は ENO，第一引数（変数 1）は EN になります。

5.1.5 ST でのファンクションブロックの呼び出し

ST でのファンクションブロックの呼び出しは，下記の記述で行います。

```
インスタンス名（入力変数 1:= 変数 1, ... 出力変数 1:= 変数 2, ...);
```

インスタンス名の後に，“()” で変数から入力変数，出力変数への代入文を囲います。
変数が複数ある場合は，各代入文の間を “,”（カンマ）で区切ります。

ファンクションブロックの実行結果は，インスタンス名の後に "."（ピリオド）を付けて出力変数を指定して，変数へ代入することにより格納されます。

1) 入力変数が 1 つ，出力変数が 1 つのファンクションブロック

FB 定義
FB 名：FBADD
FB インスタンス名：FBADD1
入力変数 1：IN1
出力変数 1：OUT1

上記のファンクションブロックを呼び出すときの記述は，下記のようになります。

```
FBADD1（IN1:=Input1）;
Output1:=FBADD1.OUT1;
```

2) 入力変数が 3 つ，出力変数が 2 つのファンクションブロック

FB 定義
FB 名：FBADD
FB インスタンス名：FBADD1
入力変数 1：IN1
入力変数 2：IN2
入力変数 3：IN3
出力変数 1：OUT1
出力変数 2：OUT2

上記のファンクションブロックを呼び出すときの記述は，下記のようになります。

```
FBADD1(IN1:=Input1, IN2:=Input2, IN3:= Input3);
Output1:=FBADD1.OUT1;
Output2:=FBADD1.OUT2;
```

ポイント

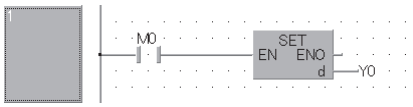
・ファンクションブロック呼び出し時の引数について
［ツール］→［オプション］→ “ コンパイル ” → “ 構造化ラダー / FBD/ST” → “ コンパイル条件 1” の “FB の呼び出し時に，VAR_OUTPUT の使用を許可する（ST プログラム）” にチェックがない場合，VAR_OUTPUT はテンプレートに表示されません。

5.1.6 条件文・反復文使用時の注意事項

ST プログラムで条件文・反復文を使用してプログラムを作成する場合の注意事項について説明します。

(1) 条件文・反復文でブール式（条件式）が一度成り立つと，＜文＞内でビットデバイスを ON している場合，そのビットデバイスは常時 ON となります。

・常時 ON となるプログラム

ST プログラム	ST プログラムと同等の処理となる 構造化ラダー /FBD プログラム
IF M0 THEN Y0 := TRUE; END_IF;	

常時 ON となるのを避けるには，以下のようにビットデバイスを OFF するプログラムを追加してください。

・常時 ON となるのを避けるプログラム例

ST プログラム *1	ST プログラムと同等の処理となる 構造化ラダー /FBD プログラム

IF M0 THEN
Y0 := TRUE;
ELSE
Y0 := FALSE;
END_IF;	

＊1：上記のプログラムは，以下のように記述することもできます。
Y0 := M0;
または
OUT(M0,Y0);
ただし，OUT 命令を条件文・反復文の＜文＞内で使用した場合は，常時 ON となるプログラムと同様の状態となります。

(2) Q00UCPU，Q00UJCPU，Q01UCPU 使用時，条件文・反復文のブール式（条件式）に文字列型を使用すると，コンパイルエラーとなる場合があります。

• コンパイルエラーとなるプログラム

ST プログラム

```
IF Var_String = "MOJI" THEN
  Y0 := TRUE;
END_IF;
```

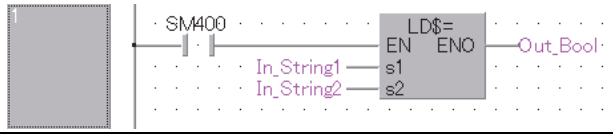
データ型に文字列型を指定すると，
コンパイルエラーになる。

コンパイルエラーとなるのを避けるには，ラダーまたは構造化ラダー /FBD で文字列型比較のファンクションブロックを作成し，ファンクションブロックの演算結果を条件文・反復文の条件式で使用してください。
下記は構造化ラダー /FBD でファンクションブロックを作成した場合の例です。

• コンパイルエラーとなるのを避けるプログラム作成例

①構造化ラダー /FBD で文字列比較のファンクションブロックを作成する。

ファンクションブロック（EQFB_01）



②ファンクションブロック（EQFB_01）の演算結果を，ST プログラムの条件式で使用する。

ラベル設定

	クラス	ラベル名	データ型	
1	VAR	Var_String	文字列(32)	...
2	VAR	Var_Bool	ビット	...
3	VAR	Inst_EQFB	EQFB_01	...

ST プログラム

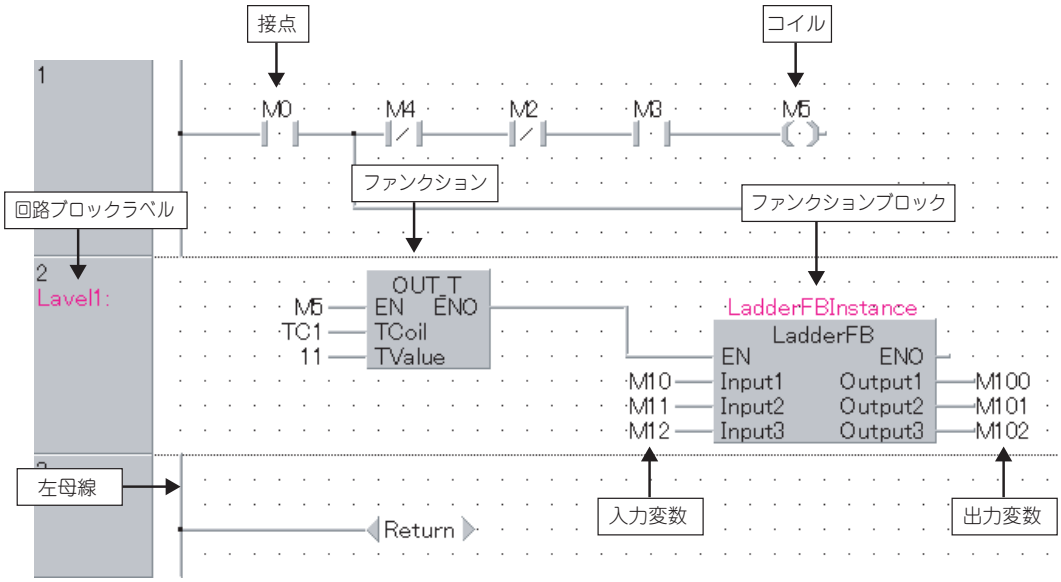
```
Inst_EQFB( In_String1 := Var_String,
           In_String2 := "MOJI",
           Out_Bool := Var_Bool );
IF TRUE = Var_Bool THEN
  Y0 := TRUE;
END_IF;
```

ファンクションブロック（EQFB_01）の演算結果を使用する。

5.2 構造化ラダー /FBD

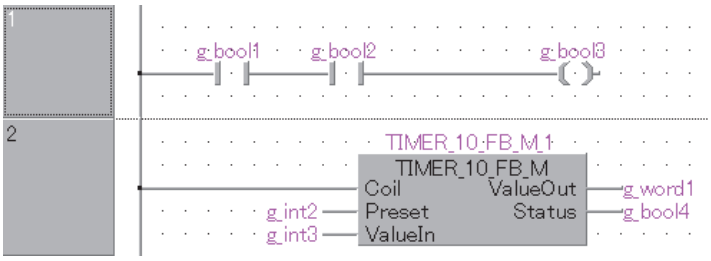
構造化ラダー /FBD は，接点，コイル，ファンクション，ファンクションブロックなどの回路記号を使用して，プログラムをグラフィカルに記述できる言語です。

5.2.1 基本書式

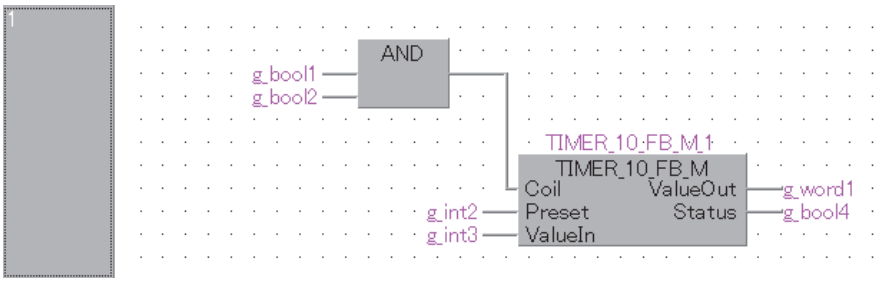


構造化ラダー /FBD でのプログラミングは，回路ブロック単位で行います。

構造化ラダーの場合，左母線から回路記号を罫線で接続します。



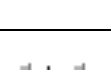
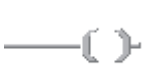
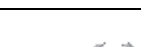
FBD の場合，左母線に接続せず，データや信号の流れに沿って回路記号を罫線で接続します。



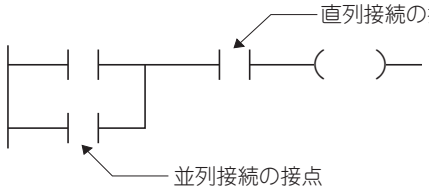
5.2.2 構造化ラダー /FBD の回路記号

構造化ラダー /FBD でのプログラミングに使用できる回路記号を下表に示します。
詳細については下記を参照してください。
☞ MELSEC-Q/L 構造化プログラミングマニュアル（共通命令編）

表 5.2.2-1 構造化ラダー /FBD の回路記号（1/2）

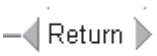
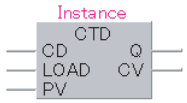
要素	回路記号	説明
a 接点 *1,*2		指定デバイスまたはラベルが ON のとき導通します。
b 接点 *1,*2		指定デバイスまたはラベルが OFF のとき導通します。
立上りパルス *1,*2,*3		指定デバイスまたはラベルの立上り時（OFF→ON）に導通します。
立下りパルス *1,*2,*3		指定デバイスまたはラベルの立下り時（ON→OFF）に導通します。
立上りパルス否定 *1,*2,*3		指定デバイスまたはラベルの OFF 時，ON 時，および立下り時（ON → OFF）に導通します。
立下りパルス否定 *1,*2,*3		指定デバイスまたはラベルの OFF 時，ON 時，および立上り時（OFF → ON）に導通します。
コイル *1		演算結果を指定デバイスまたはラベルへ出力します。
反転形コイル *1		演算結果が OFF すると指定デバイスまたはラベルが ON します。
セット *1		演算結果が ON すると指定デバイスまたはラベルが ON します。 ON したデバイスまたはラベルは，演算結果が OFF になっても ON のまま保持されます。
リセット *1		演算結果が ON すると指定デバイスまたはラベルが OFF します。演算結果が OFF の場合，デバイスまたはラベルの状態は変化しません。

*1: FBD では使用しません。
*2: 接点記号は，回路図の接続状態に応じて，AND 演算，OR 演算を行い，演算結果に反映します。
・直列接続のときは，それまでの演算結果と AND 演算を行い，演算結果とします。
・並列接続のときは，それまでの演算結果と OR 演算を行い，演算結果とします。



*3: GX Works2 のバージョン 1.11M 以降品で対応しています。
GX Works2 のバージョンの確認方法については下記を参照してください。
☞ GX Works2 Version 1 オペレーティングマニュアル（共通編）

表 5.2.2-2 構造化ラダー /FBD の回路記号（2/2）

要素	回路記号	説明
ジャンプ		ポインタ分岐命令です。 無条件に同一プログラム部品内の指令されたポインタ番号のプログラムを実行します。
リターン		サブルーチンプログラムの終了を示します。
ファンクション		ファンクションを実行します。
ファンクションブロック		ファンクションブロックを実行します。
ファンクション引数入力		ファンクション，ファンクションブロックへ引数を入力します。
ファンクション戻り値出力		ファンクション，ファンクションブロックから戻り値を出力します。
ファンクション引数反転入力		ファンクション，ファンクションブロックへ引数を反転して入力します。
ファンクション戻り値反転出力		ファンクション，ファンクションブロックからの戻り値を反転して出力します。

ポイント

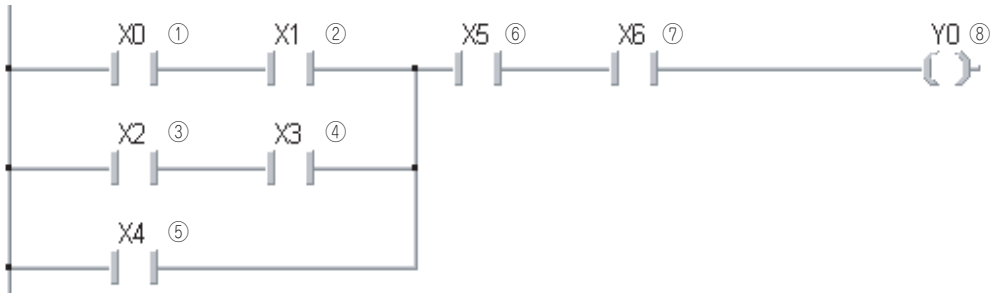
リターンは，使用するプログラムやファンクション，ファンクションブロックによって動作が異なります。

- プログラムで使用する場合
  プログラム部品の実行を終了します。
- ファンクションで使用する場合
  ファンクションを終了します。また，ファンクションをコールした命令の次のステップに戻ります。
- ファンクションブロックで使用する場合
  プロパティの「使用箇所に FB を展開」のチェックの有無により動作が異なります。
  チェックありの場合は，プログラム部品の実行を終了します。
  チェックなしの場合は，ファンクションブロックを終了します。また，ファンクションブロックをコールした命令の次のステップに戻ります。

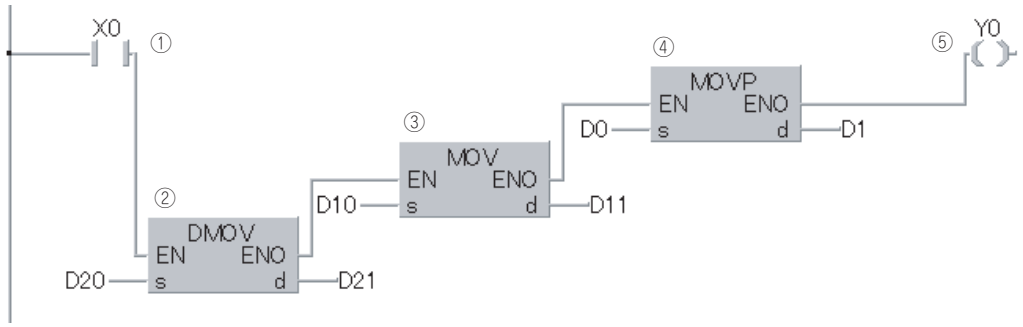
5.2.3 実行順序

プログラムの実行順序を，下記に示します。

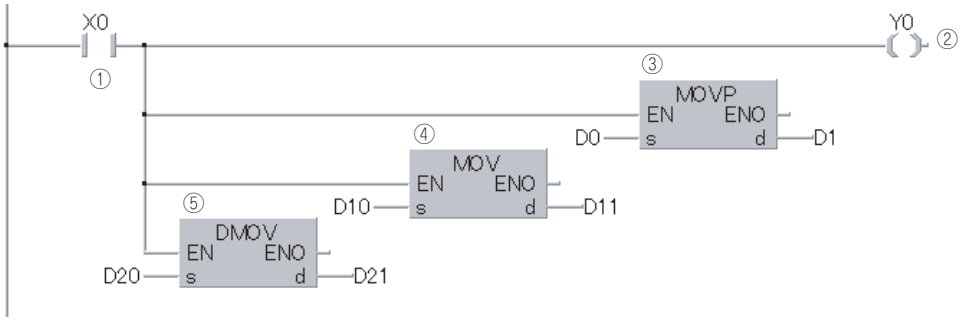
プログラムは，左から右へ，上から下へと実行します。



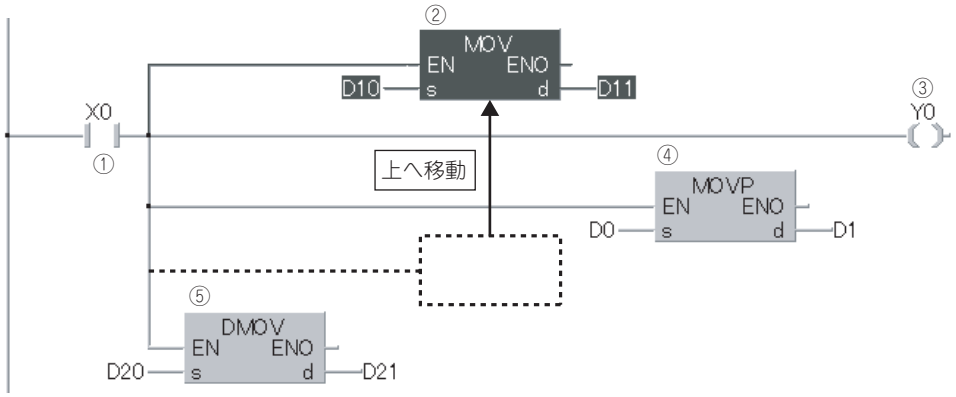
回路を分岐せず，EN/ENO を直列で接続している場合は，母線から右へ実行します。



回路分岐し，EN/ENO を並列で接続した場合は，上から下へと実行します。



上の図の，④の MOV 命令を一番上に移動した場合は，下記のようになります。



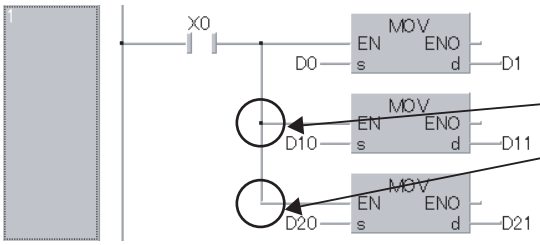
5.2.4 回路分岐におけるコンパイル結果

回路を複数に分岐した場合，分岐までのプログラムによって分岐点よりあとのプログラムのコンパイル結果が異なります。

回路分岐におけるコンパイル結果，および注意事項を下記に示します。

(1) 分岐点までの接点が 1 点の場合，コンパイル結果は接点の命令を複数回使用した状態になります。

<回路例>



<コンパイル結果>

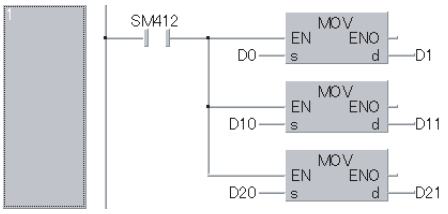
ステップ	コンパイル結果
0	LD X0
1	MOV D0 D1
3	LD X0
4	MOV D10 D11
6	LD X0
7	MOV D20 D21

接点を使用したLD命令が作成される。

＜注意事項＞

1 スキャンの最中に値が変化するデバイス (SM412 など) を使用した場合，分岐後のシーケンスプログラムの一部だけが実行され，その他のシーケンスプログラムは実行されない場合があります。

<回路例>



<コンパイル結果>

ステップ	コンパイル結果
0	LD SM412
1	MOV D0 D1
3	LD SM412
4	MOV D10 D11
6	LD SM412
7	MOV D20 D21

1スキャン中にSM412の値が変化した場合，一部のプログラムが実行されない場合がある。

1 つの接点に対して複数の命令を実行する場合は，回路を直列に接続してください。
コンパイル結果は LD 命令を一度しか使用しないシーケンスプログラムとなるため，すべてのシーケンスプログラムが実行されます。

<回路例>

<コンパイル結果>

ステップ	コンパイル結果
0	LD SM412
1	MOV D0 D1
3	MOV D10 D11
5	MOV D20 D21

(2) 分岐点までに複数の接点や，ファンクション，ファンクションブロックを使用している場合，コンパイル結果は分岐点に一時変数を付加した状態になります。

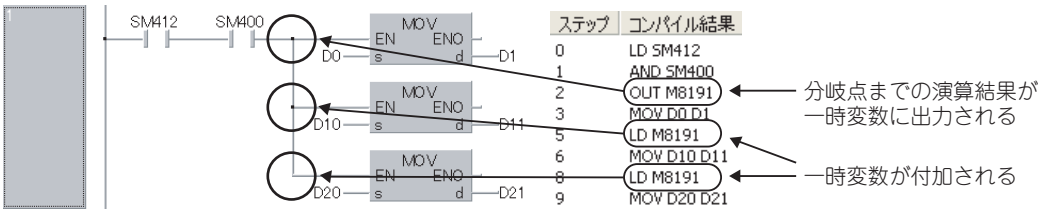
コンパイル結果に一時変数を使用しないようにする場合は，(1) の<注意事項>のように，回路を直列に接続してください。

一時変数については下記を参照してください。

☞ GX Works2 Version 1 オペレーティングマニュアル（構造化プロジェクト編）

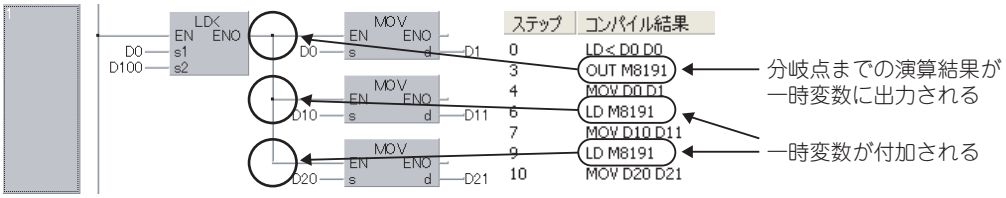
● 分岐点までに複数の接点がある場合

<回路例>　　<コンパイル結果>



● ファンクション，ファンクションブロックの出力値を分岐した場合

<回路例>　　<コンパイル結果>



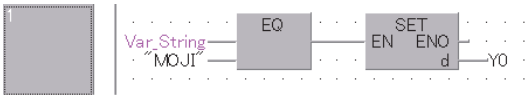
5.2.5 構造化ラダー /FBD プログラム作成時の注意事項

構造化ラダー /FBD でプログラムを作成する場合の注意事項について説明します。

Q00UCPU，Q00UJCPU，Q01UCPU 使用時，比較関数の入力に文字列型を使用すると，コンパイルエラーとなる場合があります。

・ コンパイルエラーとなるプログラム例

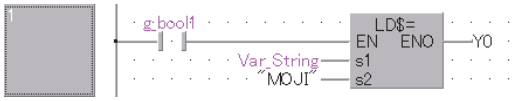
構造化ラダー /FBD プログラム



コンパイルエラーとなるのを避けるには，LD$=, LD$<>, LD$<=, LD$<, LD$>=, LD$> 命令を使用してください。

・ コンパイルエラーを避けるプログラム例

構造化ラダー /FBD プログラム



メ　モ

付　録

付 .1　普遍的データ型に対応するデバイス . . . . . 付 -2
付 .2　ラベル名やデータ名で使用できない文字列 . . . . . 付 -6
付 .3　ラダーからの置換え . . . . . 付 -9

付.1 普遍的データ型に対応するデバイス

普遍的データ型に対応するデバイスを下記に示します。

表 付.1-1 普遍的データ型に対応するデバイス

デバイス				
分類	種別	デバイス名	デバイスシンボル	
内部ユーザデバイス	ビットデバイス	入力	X	
		出力	Y	
		内部リレー	M	
		ラッチリレー	L	
		アナンシエータ	F	
		エッジリレー	V	
		ステップリレー	S	
		リンク特殊リレー	SB	
		リンクリレー	B	
		タイマ接点 *1	TS	
		タイマコイル *1	TC	
		積算タイマ接点 *1	STS	
		積算タイマコイル *1	STC	
		カウンタ接点 *1	CS	
		カウンタコイル	CC	
	ワードデバイス	タイマ現在値	T または TN*1	
		積算タイマ現在値	ST または STN*1	
		カウンタ現在値	C または CN*1	
		データレジスタ	D	
		リンクレジスタ	W	
		リンク特殊レジスタ	SW	
内部システムデバイス	ビットデバイス	ファンクション入力	FX	
		ファンクション出力	FY	
		特殊リレー	SM	
	ワードデバイス	ファンクションレジスタ	FD	
		特殊レジスタ	SD	

*1： 桁指定時に使用可能
*2： ビット指定時に使用可能

	普遍的データ型											
	ANY										ANY	
	ANY_SIMPLE											
	ANY_BIT			ANY_NUM				時間	文字列	配列	構造体	
				ANY_INT		ANY_REAL						
	ビット	ワード[符号なし]/ビット列[16ビット]	ダブルワード[符号なし]/ビット列[32ビット]	ワード[符号付き]	ダブルワード[符号付き]	単精度実数	倍精度実数					ANY16	ANY32
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	×	×	×	×	×	×	×	×	×	×	×	×
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	×	○	×	○	×	×	×	×	×	×	×	○	×
	×	○	×	○	×	×	×	×	×	×	×	○	×
	×	○	×	○	×	×	×	×	×	×	×	○	×
	○*2	○	×	○	×	×	×	×	×	×	×	○	×
	○*2	○	×	○	×	×	×	×	×	×	×	○	×
	○*2	○	×	○	×	×	×	×	×	×	×	○	×
	−	−	−	−	−	−	−	−	−	−	−	−	−
	−	−	−	−	−	−	−	−	−	−	−	−	−
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	−	−	−	−	−	×	×	×	×	×	×		−
	○*2	○	×	○	×	×	×	×	×	×	×	○	×

付

付 録

デバイス				
分類	種別	デバイス名	デバイスシンボル	
リンクダイレクトデバイス	ビットデバイス	リンク入力	Jn¥X	
		リンク出力	Jn¥Y	
		リンクリレー	Jn¥B	
		リンク特殊リレー	Jn¥SB	
	ワードデバイス	リンクレジスタ	Jn¥W	
		リンク特殊レジスタ	Jn¥SW	
インテリジェント機能ユニットデバイス	ワードデバイス	インテリジェント機能ユニットデバイス	Un¥G	
インデックスレジスタ	ワードデバイス	インデックスレジスタ	Z	
ファイルレジスタ	ワードデバイス	ファイルレジスタ	R または ZR	
ネスティング	−	ネスティング	N	
ポインタ	−	ポインタ	P	
		割込みポインタ	I	
定数	−	−	K, H	
			E	
文字列型定数	−	−	‘文字列’ または “文字列”	

*1： 桁指定時に使用可能
*2： ビット指定時に使用可能

	普遍的データ型											
	ANY										ANY	
	ANY_SIMPLE											
	ANY_BIT			ANY_NUM				時間	文字列	配列	構造体	
				ANY_INT		ANY_REAL						
	ビット	ワード[符号なし]/ビット列[16ビット]	ダブルワード[符号なし]/ビット列[32ビット]	ワード[符号付き]	ダブルワード[符号付き]	単精度実数	倍精度実数					ANY16	ANY32
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○*2	○	×	○	×	×	×	×	×	×	×	○	×
	○*2	○	×	○	×	×	×	×	×	×	×	○	×
	○*2	○	×	○	×	×	×	×	×	×	×	○	×
	×	○	×	○	×	×	×	×	×	×	×	○	×
	○*2	○	×	○	×	×	×	×	×	×	×	○	×
	−	−	−	−	−	−	−	−	−	−	−	−	−
	−	−	−	−	−	−	−	−	−	−	−	−	−
	−	−	−	−	−	−	−	−	−	−	−	−	−
	○	○	○	○	○	○	×	×	×	×	×	○	○
	×	×	×	×	×	○	○	×	×	×	×	×	×
	×	×	×	×	×	×	×	×	○	×	×	×	×

付

付 録

付 .2 ラベル名やデータ名で使用できない文字列

応用関数名，共通命令名，特殊命令名，命令語などで使用している文字列を，予約語と呼びます。

予約語はラベル名やデータ名として使用できません。予約語として定義されている文字列をラベル名やデータ名で使用した場合は，登録／コンパイルを実行したときにエラーになります。

下記に，ラベル名やデータ名として使用できない文字列を示します。

表内の①～⑨の項目は，以下に示すラベル名およびデータ名の種別と対応しています。

＜ラベル名およびデータ名の種別＞

①プロジェクトファイル名
②プログラムファイル名 (シンプル (ラベルを使用しない))
③プログラムファイル名 (シンプル (ラベルを使用する))
④プログラムファイル名 (構造化)
⑤タスク名
⑥グローバルラベルのデータ名
⑦構造体名
⑧プログラム部品名
⑨ラベル名

表 付 .2-1 ラベル名やデータ名で使用できない文字列 (1/3)

カテゴリ	文字列	①	②	③	④	⑤	⑥	⑦	⑧	⑨
クラス識別子	VAR, VAR_RETAIN, VAR_ACCESS, VAR_CONSTANT, VAR_CONSTANT_RETAIN, VAR_INPUT, VAR_INPUT_RETAIN, VAR_OUTPUT, VAR_OUTPUT_RETAIN, VAR_IN_OUT, VAR_IN_EXT, VAR_EXTERNAL, VAR_EXTERNAL_CONSTANT, VAR_EXTERNAL_CONSTANT_RETAIN, VAR_EXTERNAL_RETAIN, VAR_GLOBAL, VAR_GLOBAL_CONSTANT, VAR_GLOBAL_CONSTANT_RETAIN, VAR_GLOBAL_RETAIN	×	○	×	×	×	×	×	×	×
データ型	BOOL, BYTE, INT, SINT, DINT, LINT, UINT, USINT, UDINT, ULINT, WORD, DWORD, LWORD, ARRAY, REAL, LREAL, TIME, STRING, TIMER, COUNTER, RETENTIVETIMER, POINTER, ビット，ワード [符号なし]/ ビット列 [16 ビット]，ダブルワード [符号なし]/ ビット列 [32 ビット]，ワード [符号付き]，ダブルワード [符号付き]，単精度実数，倍精度実数，文字列，時間，タイマ，カウンタ，積算タイマ，ポインタ	○	○	×	×	×	×	×	×	×
データタイプ階層	ANY, ANY_NUM, ANY_BIT, ANY_REAL, ANY_INT, ANY_DATE	○	○	×	×	×	×	×	×	×
	ANY_SIMPLE, ANY16, ANY32	○	○	○	○	○	○	○	△ *1	×
デバイス名	X, Y, D, M, T, B, C, F, L, P, V, Z, W, I, N, U, J, K, H, E, A, SD, SM, SW, SB, FX, FY, DX, DY, FD, TR, BL, SG, VD, ZR, ZZ*2	○	○	○	○	○	○	○	△ *1	×
デバイスとして認識される文字列 (デバイス名＋数字)	X0 など	○	○	×	×	×	×	×	△ *3	×
ST 演算子	NOT, MOD	○	○	×	×	×	×	×	×	×
	(,), -	○	○	○	○	○	○	○	△ *1	×

○：使用可，△：制約事項あり，×：使用不可

*1： ファンクションは使用できません。
*2： ZZ デバイスで修飾されたデバイス名を予約語扱いとするかどうかは，パラメータの設定によります。
32 ビットインデックス修飾設定で，Z デバイスを指定した場合：予約語扱いとしません。
32 ビットインデックス修飾設定で，ZZ デバイスを指定した場合：予約語扱いとなります。
*3： シンプルプロジェクト（ラベルを使用しない）時のみ使用できます。

表 付 .2-2 ラベル名やデータ名で使用できない文字列 (2/3)

カテゴリ	文字列	①	②	③	④	⑤	⑥	⑦	⑧	⑨
IL 演算子	LD, LDN, ST, STN, S, S1, R, R1, AND, ANDN, OR, ORN, XOR, XORN, ADD, SUB, MUL, DIV, GT, GE, EQ, NE, LE, LT, JMP, JMPC, JMPCN, CAL, CALC, CALCN, RET, RETC, RETCN	×	○	×	×	×	×	×	×	×
	LDI, LDP, LDPI, LDF, LDFI, ANI, ANDP, ANDPI, ANDF, ANDFI, ANB, ORI, ORP, ORPI, ORF, ORFI, ORB, MPS, MRD, MPP, INV, MEP, MEF, EGP, EGF, OUT(H), SET, RST, PLS, PLF, FF, DELTA(P), SFT(P), MC, MCR, STOP, PAGE, NOP, NOPLF	○	○	○	○	○	○	○	△ *1	×
GX Works2 における応用命令	DMOD, PCHK, INC(P) などの応用命令 ☞ MELSEC-Q/L プログラミングマニュアル（共通命令編）, MELSEC-Q/L 構造化プログラミングマニュアル（共通命令編） ☞ FXCPU 構造化プログラミングマニュアル [シーケンス命令編], FXCPU 構造化プログラミングマニュアル [応用関数編]	○	○	○	○	○	○	○	△ *1	×
SFC 命令	SFCP, SFCPEND, BLOCK, BEND, TRANL, TRANO, TRANA, TRANC, TRANCA, TRANOA, SEND, TRANOC, TRANOCA, TRANCO, TRANCOC, STEPN, STEPD, STEPSC, STEPSE, STEPST, STEPR, STEPC, STEPG, STEPI, STEPID, STEPISC, STEPISE, STEPIST, STEPIR, TRANJ, TRANOJ, TRANOCJ, TRANCJ, TRANCOJ, TRANCOCJ	○	○	○	○	○	○	○	△ *1	×
ST コードボディ	RETURN, IF, THEN, ELSE, ELSIF, END_IF, CASE, OF, END_CASE, FOR, TO, BY, DO, END_FOR, WHILE, END_WHILE, REPEAT, UNTIL, END_REPEAT, EXIT, TYPE, END_TYPE, STRUCT, END_STRUCT, RETAIN, VAR_ACCESS, END_VAR, FUNCTION, END_FUNCTION, FUNCTION_BLOCK, END_FUNCTION_BLOCK, STEP, INITIAL_STEP, END_STEP, TRANSITION, END_TRANSITION , FROM, TO, UNTILWHILE	○	○	×	×	×	×	×	×	×
応用関数のファンクション名	AND_E, NOT_E などの応用関数のファンクション名	○	○	○	○	○	○	×	×	×
応用関数のファンクションブロック名	CTD, CTU などの応用関数のファンクションブロック名	○	○	○	○	○	○	×	×	×
記号	/, \, *, ?, <, >, ¦, ", :, [,], ., =, +, %, ', ~, @, {, }, &, ^, ., ', タブ文字	×	×	×	×	×	×	×	×	×
	;	○	×	×	×	×	×	×	×	×
	!, #, $, `	○	○	○	○	×	○	×	×	×
日付時刻リテラル	DATE, DATE_AND_TIME, DT, TIME, TIME_OF_DAY, TOD	×	○	×	×	×	×	×	×	×
その他	ACTION, END_ACTION, CONFIGURATION, END_CONFIGURATION, CONSTANT, F_EDGE, R_EDGE, AT, PROGRAM, WITH, END_PROGRAM, TRUE, FALSE, READ_ONLY, READ_WRITE, RESOURCE, END_RESOURCE, ON, TASK, EN, ENO, BODY_CCE, BODY_FBD, BODY_IL, BODY_LD, BODY_SFC, BODY_ST, END_BODY, END_PARAMETER_SECTION, PARAM_FILE_PATH, PARAMETER_SECTION, SINGLE, RETAIN, INTERVAL	×	○	×	×	×	×	×	×	×

○：使用可，△：制約事項あり，×：使用不可

*1： ファンクションは使用できません。

表 付 .2-1 ラベル名やデータ名で使用できない文字列 (3/3)

カテゴリ	文字列	①	②	③	④	⑤	⑥	⑦	⑧	⑨
K1 ～ K8 で始まる文字列	K1AAA など	○	○	○	○	○	○	○	△ *1	×
アドレス	%IX0 など	○	×	×	×	×	×	×	×	×
ラダー言語でのステートメント	:FB BLK START, :FB START, :FB END, :FB BLK END, :FB IN, :FB OUT, :FB_NAME;,INSTANCE_NAME, :FB, :INSTANCE	○	×	×	×	×	×	×	×	×
共通命令	MOV など	○	○	×	○	○	×	×	△ *3	×
Windows 予約語	COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, LPT9, AUX, CON, PRN, NUL	×	×	×	×	×	×	×	×	○

○：使用可，△：制約事項あり，×：使用不可

*1： ファンクションは使用できません。
*3： シンプルプロジェクト（ラベルを使用しない）時のみ使用できます。

(1) その他 , ラベル使用時の注意事項

- ファンクション内で，ファンクションと同一のラベル名は使用できません。
- ラベル名と，ファンクションブロック名や構造体名などのデータ名では，使用できない文字列が異なります。そのため，インスタンスや構造体型のラベルを設定するときに，ファンクションブロック名や構造体名を含んだラベル名を設定すると，エラーになる場合があります。
- スペースは使用できません。
- 先頭文字に半角の数字は使用できません。
- ラベル名の大文字，小文字は区別されません。大文字，小文字の違いがあるものの同じラベル名（例：“AAA” と “aaa”）を宣言した場合，コンパイル時にエラーとなります。
- 構造化ラダー/FBDおよびSTでは，GX Works2の下記オプション設定*1によりグローバルラベルとローカルラベルに同一ラベル名を使用できます。
  *1: [ツール]⇒[オプション]⇒ “コンパイル” ⇒ “基本設定” で “グローバルラベルとローカルラベルに同一ラベル名を使用する”
- _(アンダーバー) は，ラベル名の先頭および最後には使用できません。
  また，連続する _(アンダーバー) は，データ名およびラベル名には使用できません。
- シンプルプロジェクトの場合は，共通命令，応用関数のファンクション名，ファンクションブロック名を使用できます。

付.3 ラダーからの置換え

プログラム言語のラダーで作成したプログラムと同じ内容の構造化プログラムを，GX Works2 で作成するときの例を示します。

付.3.1 作成手順

プログラム言語のラダーで作成したプログラムをもとに，構造化プログラムを作成するときの基本的な手順を示します。

1. デバイスからラベルへの置換え

手順
ラベルには，グローバルラベルとローカルラベルがあります。 デバイスをラベルに置き換えるときの，ラベルの種類（グローバルラベル，ローカルラベル）を決めます。

2. ラベルの設定

手順
プログラムで使用するグローバルラベル，ローカルラベルは，定義が必要です。 プログラムで使用するすべてのラベルを定義します。

3. プログラムの作成

手順
使用するプログラム言語で構造化プログラムを作成します。

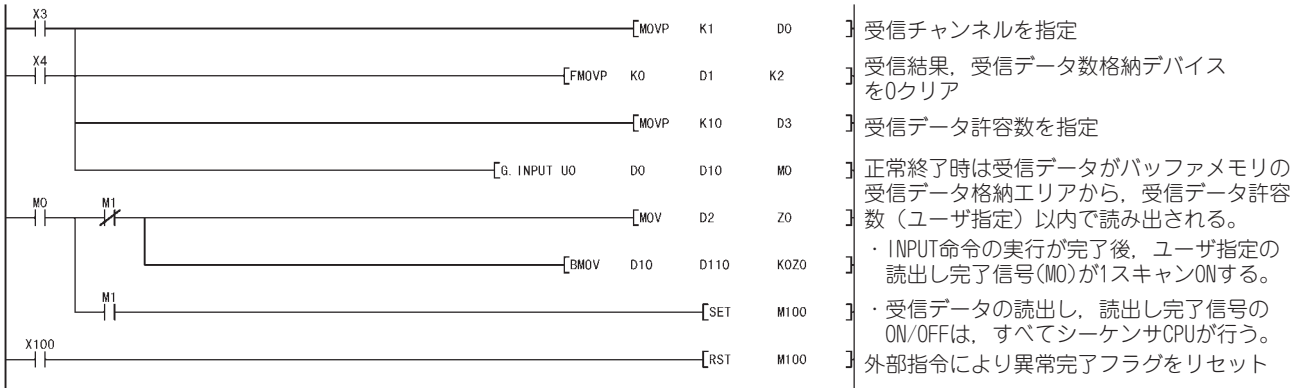
付
付録

付 .3.2 作成例

GX Developer で作成したシーケンスプログラムと同じ内容の構造化プログラムを，GX Works2 で作成するときの例を示します。

作成例として，Q 対応シリアルコミュニケーションユニットのデータ受信プログラムと同じ内容の構造化プログラムを，プログラム言語の構造化ラダー /FBD および ST で作成する例を示します。

下記に，作成もとのプログラムを示します。



(1) デバイスからラベルへの置換え

使用しているデバイスをラベルに置き換えます。
入出力デバイスなどは，グローバルラベルに置き換えます。内部リレーなどはローカルラベルに置き換えます。

表 付 3.2-1 デバイスからラベルへの置換え例

デバイス	用途		ラベル	
			データ型	ラベル名
X3	CH1 受信読出し要求		ビット	CH1ReadRequest
X4	CH1 受信異常検出		ビット	CH1AbnormalDetection
D0	コントロールデータ	受信チャンネル	ワード [符号なし]/ビット列 [16 ビット] [0] ～ [3]	ControlData
D1		受信結果		
D2		受信データ数		
D3		受信データ許容数		
D10 ～ D109	受信データ		ワード [符号なし]/ビット列 [16 ビット] [0] ～ [99]	ReceiveData
D110 ～ D209	受信データ格納領域		ワード [符号なし]/ビット列 [16 ビット] [0] ～ [99]	Data
M0	受信完了フラグ	完了フラグ	ビット [0] ～ [1]	Completion
M1		完了時の状態フラグ		
M100	異常完了フラグ		ビット	AbnormalCompletion
X100	異常完了フラグリセット指令		ビット	ResetAbnormalCompletion

(2) ラベルの設定

グローバルラベルとローカルラベルの設定を行います。

• グローバルラベルの設定例

		クラス	ラベル名	データ型		定数値	デバイス	アドレス
	1	VAR_GLOBAL	CH1_ReadRequest	ビット	...		X3	%IX3
	2	VAR_GLOBAL	CH1_AbnormalDetection	ビット	...		X4	%IX4
	3	VAR_GLOBAL	ResetAbnormalCompletion	ビット	...		X100	%IX256

• ローカルラベルの設定例*1

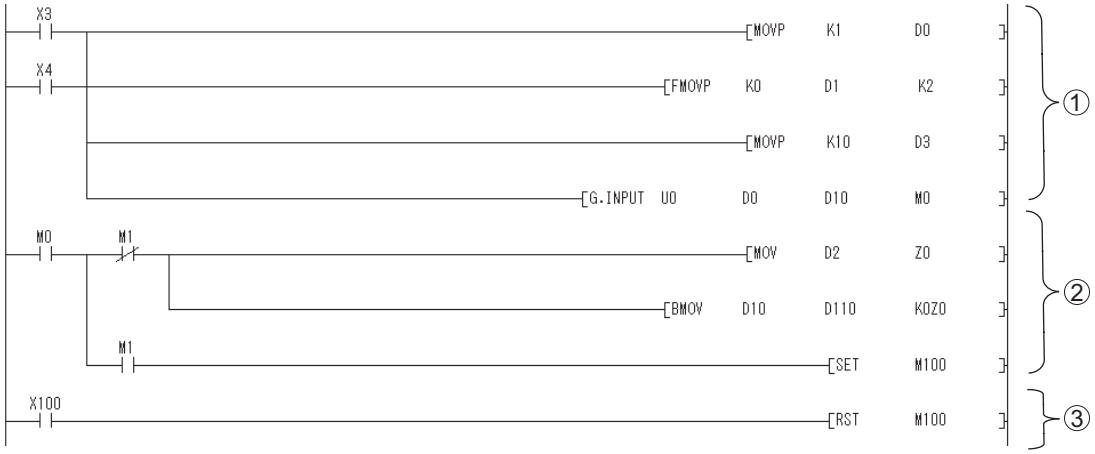
		クラス	ラベル名	データ型		定数値
	1	VAR	ControlData	ワード[符号なし]/ビット列[16ビット](0..3)	...	
	2	VAR	ReceiveData	ワード[符号なし]/ビット列[16ビット](0..1)	...	
	3	VAR	Completion	ビット(0..1)	...	
	4	VAR	Data	ワード[符号なし]/ビット列[16ビット](0..9)	...	
	5	VAR	AbnormalCompletion	ビット	...	

*1： ローカルラベルのデバイスは，GX Works2 の自動割付デバイス設定で指定された範囲が自動で割り付けられます。
もとのラダープログラムと同じデバイスを割り付ける場合は，グローバルラベルで設定してください。

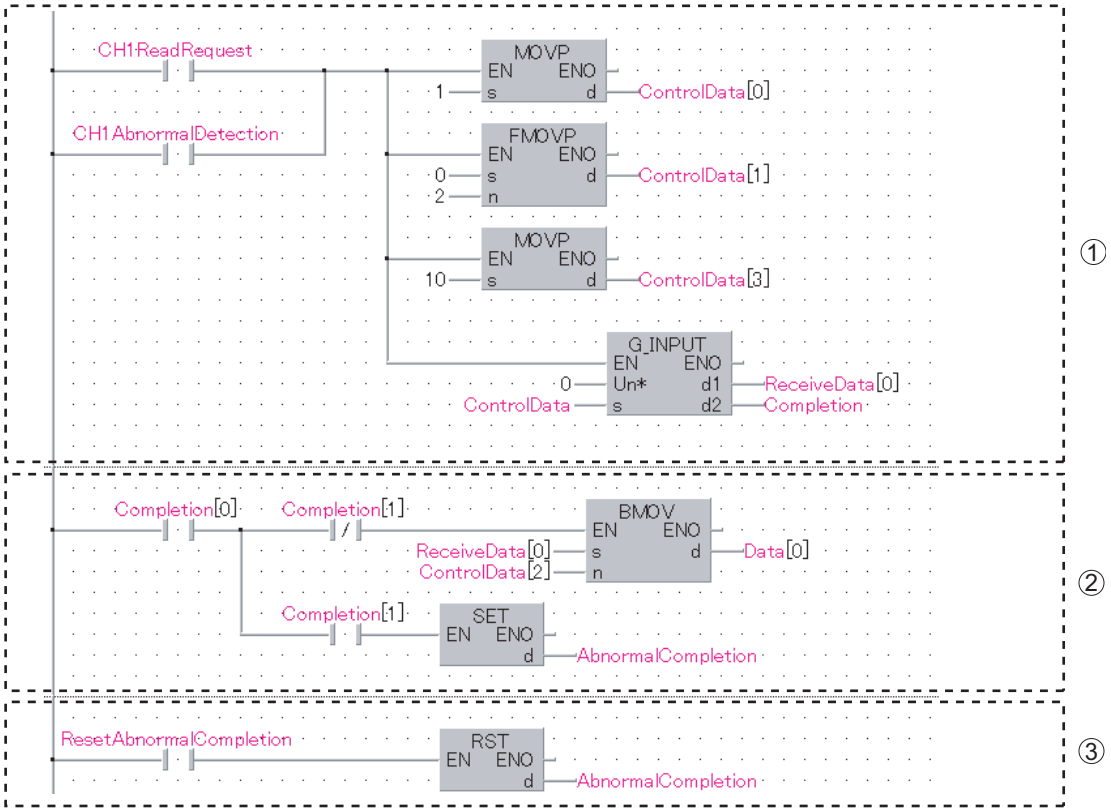
(3) 構造化プログラムの作成

作成もとのプログラムから，構造化プログラムを作成します。

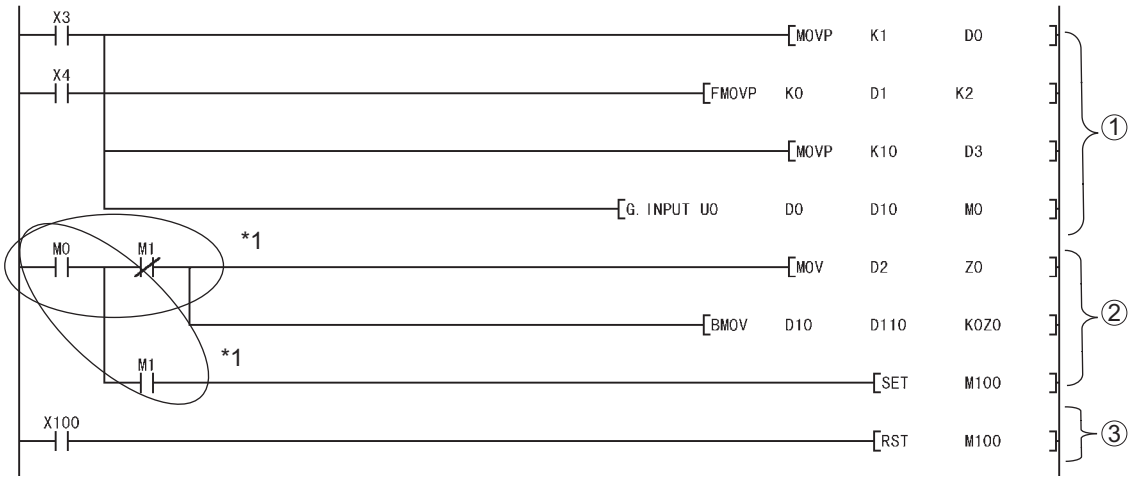
• 作成もとのプログラム（プログラム言語：ラダー）



• 構造化プログラム（プログラム言語：構造化ラダー /FBD）



• 作成もとのプログラム（プログラム言語：ラダー）



• 構造化プログラム（プログラム言語：ST）

```
IF CH1ReadRequest OR CH1AbnormalDetection THEN

    ControlData[0] := 1;
    ControlData[1] := 0;
    ControlData[2] := 0;
    ControlData[3] := 10;

    G_INPUT(TRUE, 0, ControlData, ReceiveData[0], Completion);

END_IF;
```

①

```
BMOV(Completion[0] AND NOT Completion[1], ReceiveData[0], ControlData[2], Data[0]);
SET(Completion[0] AND Completion[1], AbnormalCompletion);
```

*1

②

```
RST(ResetAbnormalCompletion, AbnormalCompletion);
```

③

*1: 実行条件の接点が複数ある場合，“()” で囲み，複数の接点を一括してプログラミングすることができます。

付 付録

メ　モ

索　引

【E】
EN 4-14
ENO 4-14
【F】
FBD 5-13
【S】
ST 4-10
【ア】
アドレス 4-40, 4-41
【イ】
インスタンス 4-7, 4-13
インデックス修飾 4-44
【エ】
演算子 5-3
【カ】
階層化 1-7, 2-2
回路記号 5-14
回路ブロック 4-9
回路ブロックラベル 4-9
【キ】
基本書式 5-2, 5-13
基本データ型 4-19
【ク】
クラス 4-17
グローバルラベル 4-16
【コ】
構造化設計 1-7
構造化ラダー 5-13
構造化ラダー /FBD 5-13
構造体 4-38
構文 5-4
【サ】
32 ビットによるインデックス修飾 4-46
【シ】
実行条件 4-4
出力変数 4-11, 4-17
【タ】
タスク 4-4
ダブルワード（32 ビット）データ 4-26
単精度実数データ 4-29
【テ】
定数 4-21
データ型 4-19
データの指定方法 4-22
デバイス ..... 4-39, 4-41, 付 -2
【ニ】
入出力変数 4-17
入力変数 4-17
【ハ】
倍精度実数データ 4-30
配列 4-35
【ヒ】
ビットデータ 4-23
ビットデータの桁指定 4-24
【フ】
ファンクション 4-6
ファンクションの呼び出し 5-9
ファンクションブロック 4-7
ファンクションブロックの呼び出し 5-10
部品化 1-7, 2-3
普遍的データ型 4-20
普遍的データ型に対応するデバイス ..... 付 -2
プログラム 4-5
プログラムファイル 4-3
プログラム部品 4-5
プログラムブロック 4-6
プロジェクト 2-2, 4-3
【モ】
文字列データ 4-33
【ユ】
ユーザライブラリ 4-57
優先度 4-4
【ラ】
ライブラリ 4-56
ラベル名称の注意事項 4-58
【ロ】
ローカルラベル 4-16
【ワ】
ワードデバイスのビット指定 4-23
ワード（16 ビット）データ 4-24

保証について

ご使用に際しましては，以下の製品保証内容をご確認いただきますよう，よろしくお願いいたします。

１．無償保証期間と無償保証範囲

無償保証期間中に，製品に当社側の責任による故障や瑕疵（以下併せて「故障」と呼びます）が発生した場合，当社はお買い上げいただきました販売店または当社サービス会社を通じて，無償で製品を修理させていただきます。ただし，国内および海外における出張修理が必要な場合は，技術者派遣に要する実費を申し受けます。
また，故障ユニットの取替えに伴う現地再調整・試運転は当社責務外とさせていただきます。

【無償保証期間】

製品の無償保証期間は，お客様にてご購入後またはご指定場所に納入後 36ヵ月とさせていただきます。
ただし，当社製品出荷後の流通期間を最長 6ヵ月として，製造から 42ヵ月を無償保証期間の上限とさせていただきます。また，修理品の無償保証期間は，修理前の無償保証期間を超えて長くなることはありません。

【無償保証範囲】

(1) 一次故障診断は，原則として貴社にて実施をお願い致します。
ただし，貴社要請により当社，または当社サービス網がこの業務を有償にて代行することができます。この場合，故障原因が当社側にある場合は無償と致します。

(2) 使用状態・使用方法，および使用環境などが，取扱説明書，ユーザーズマニュアル，製品本体注意ラベルなどに記載された条件・注意事項などに従った正常な状態で使用されている場合に限定させていただきます。

(3) 無償保証期間内であっても，以下の場合には有償修理とさせていただきます。
①お客様における不適切な保管や取扱い，不注意，過失などにより生じた故障およびお客様のハードウェアまたはソフトウェア設計内容に起因した故障。
②お客様にて当社の了解なく製品に改造などの手を加えたことに起因する故障。
③当社製品がお客様の機器に組み込まれて使用された場合，お客様の機器が受けている法的規制による安全装置または業界の通念上備えられているべきと判断される機能・構造などを備えていれば回避できたと認められる故障。
④取扱説明書などに指定された消耗部品が正常に保守・交換されていれば防げたと認められる故障。
⑤消耗部品（バッテリ，リレー，ヒューズなど）の交換。
⑥火災，異常電圧などの不可抗力による外部要因および地震，雷，風水害などの天変地異による故障。
⑦当社出荷当時の科学技術の水準では予見できなかった事由による故障。
⑧その他，当社の責任外の場合またはお客様が当社責任外と認めた故障。

２．生産中止後の有償修理期間

(1) 当社が有償にて製品修理を受け付けることができる期間は，その製品の生産中止後 7 年間です。
生産中止に関しましては，当社テクニカルニュースなどにて報じさせていただきます。

(2) 生産中止後の製品供給（補用品も含む）はできません。

３．海外でのサービス

海外においては，当社の各地域 FA センターで修理受付をさせていただきます。ただし，各 FA センターでの修理条件などが異なる場合がありますのでご了承ください。

４．機会損失，二次損失などへの保証責務の除外

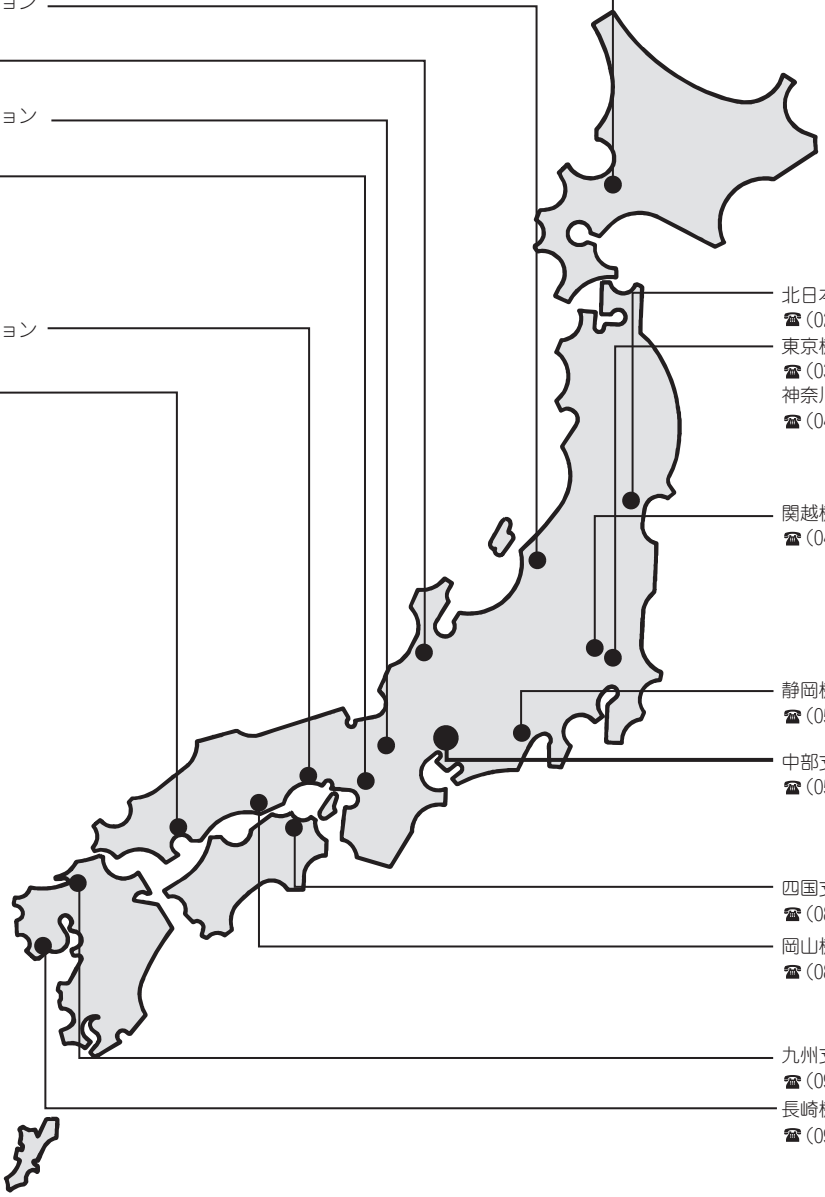
無償保証期間の内外を問わず，当社の責に帰すことができない事由から生じた障害，当社製品の故障に起因するお客様での機会損失，逸失利益，当社の予見の有無を問わず特別の事情から生じた損害，二次損害，事故補償，当社製品以外への損傷，およびお客様による交換作業，現地機械設備の再調整，立上げ試運転その他の業務に対する補償については，当社責務外とさせていただきます。

５．製品仕様の変更

カタログ，マニュアルもしくは技術資料などに記載の仕様は，お断りなしに変更させていただく場合がありますので，あらかじめご承知おきください。

以　上

サービスネットワーク（三菱電機システムサービス株式会社）



Microsoft，Windows，Windows Vista，Windows NT，Windows XP，Windows Server，Visio，Excel，PowerPoint，Visual Basic，Visual C++，Access は，米国 Microsoft Corporation の米国，日本およびその他の国における登録商標または商標です。
Intel，Pentium，Celeron は米国およびその他の国における Intel Corporation の商標です。
イーサネット，Ethernet は富士ゼロックス株式会社の登録商標です。
PC-9800 は日本電気株式会社の商標です。
PC98-NX は NEC パーソナルコンピュータ株式会社の登録商標です。
SD ロゴ，SDHC ロゴは SD-3C，LLC の登録商標または商標です。
その他の製品名，社名はそれぞれの会社の商標，または登録商標です。

SH(名)-080735-R

MELSEC-Q/L/F
構造化プログラミングマニュアル
基礎編

三菱電機株式会社 〒100-8310 東京都千代田区丸の内2-7-3(東京ビル)

お問い合わせは下記へどうぞ

本社機器営業部 …… 〒100-8310 東京都千代田区丸の内2-7-3(東京ビル) …… (03)3218-6760
北海道支社 …… 〒060-8693 札幌市中央区北二条西4-1(北海道ビル) …… (011)212-3794
東北支社 …… 〒980-0011 仙台市青葉区上杉1-17-7(仙台上杉ビル) …… (022)216-4546
関越支社 …… 〒330-6034 さいたま市中央区新都心11-2(明治安田生命さいたま新都心ビル) …… (048)600-5835
新潟支店 …… 〒950-8504 新潟市中央区東大通2-4-10(日本生命ビル) …… (025)241-7227
神奈川支社 …… 〒220-8118 横浜市西区みなとみらい2-2-1(横浜ランドマークタワー) …… (045)224-2624
北陸支社 …… 〒920-0031 金沢市広岡3-1-1(金沢パークビル) …… (076)233-5502
中部支社 …… 〒451-8522 名古屋市西区牛島町6-1(名古屋ルーセントタワー) …… (052)565-3314
豊田支店 …… 〒471-0034 豊田市小坂本町1-5-10(矢作豊田ビル) …… (0565)34-4112
関西支社 …… 〒530-8206 大阪市北区堂島2-2-2(近鉄堂島ビル) …… (06)6347-2771
中国支社 …… 〒730-8657 広島市中区中町7-32(ニッセイ広島ビル) …… (082)248-5348
四国支社 …… 〒760-8654 高松市寿町1-1-8(日本生命高松駅前ビル) …… (087)825-0055
九州支社 …… 〒810-8686 福岡市中央区天神2-12-1(天神ビル) …… (092)721-2247

三菱 FA 検索

www.MitsubishiElectric.co.jp/fa/

メンバー登録無料!

インターネットによる情報サービス「三菱電機FAサイト」

三菱電機FAサイトでは、製品や事例などの技術情報に加え、トレーニングスクール情報や各種お問い合わせ窓口をご提供しています。また、メンバー登録いただくとマニュアルやCADデータ等のダウンロード、eラーニングなどの各種サービスをご利用いただけます。

三菱電機FA機器電話, FAX技術相談

●電話技術相談窓口 受付時間※1 月曜～金曜 9:00～19:00、土曜・日曜・祝日 9:00～17:00

対象機種			電話番号
シーケンサ	MELSEC-Q/L/QnA/Aシーケンサー般(下記以外)		052-711-5111
	MELSEC-F FX/Fシーケンサ全般		052-725-2271※2
	ネットワークユニット/シリアルコミュニケーションユニット		052-712-2578
	アナログユニット/温調ユニット/温度入力ユニット/高速カウンタユニット		052-712-2579
	MELSOFT シーケンサプログラミングツール	MELSOFT GXシリーズ SW□IVD-GPPA/GPPQなど	052-711-0037
	MELSOFT 統合エンジニアリング環境	MELSOFT iQ Works(Navigator)	052-712-2370
	MELSOFT 通信支援ソフトウェアツール	MELSOFT MXシリーズ SW□D5F-CSKP/OLEX/XMOPなど	
	MELSECパソコンボード	Q80BDシリーズなど	
	C言語コントローラ/MESインタフェースユニット/高速データロガーユニット		
	iQ Sensor Solution		
	MELSEC計装/Q二重化	プロセスCPU 二重化CPU MELSOFT PXシリーズ	052-712-2830※2
	MELSEC Safety	安全シーケンサ(MELSEC-QSシリーズ) 安全コントローラ(MELSEC-WSシリーズ)	052-712-3079※2
	電力計測/絶縁監視ユニット	QE8□シリーズ	052-719-4557※2※3
表示器		GOT-F900/DUシリーズ	052-725-2271※2
		GOT1000/A900シリーズなど MELSOFT GTシリーズ	052-712-2417
サーボ/位置決めユニット/モーションコントローラ		MELSERVOシリーズ 位置決めユニット/シンプルモーションユニット モーションCPU(Q/Aシリーズ) MELSOFT MTシリーズ/MRシリーズ	052-712-6607
インバータ		FREQROLシリーズ	052-722-2182
ロボット		MELFAシリーズ	052-721-0100

※1:春季・夏季・年末年始の休日を除く ※2:金曜は17:00まで ※3:土曜・日曜・祝日を除く

●FAX技術相談窓口 受付時間※4 9:00～16:00(受信は常時※5)

対象機種	FAX番号
上記電話技術相談対象機種	052-719-6762
電力計測/絶縁監視ユニット(QE8□シリーズ)	084-926-8340

三菱電機FAサイトの「仕様・機能に関するお問い合わせ」もご利用ください。
※4:土曜・日曜・祝日、春季・夏季・年末年始の休日を除く ※5:春季・夏季・年末年始の休日を除く

本マニュアルは, 輸出する場合, 経済産業省への役務取引許可申請は不要です。

形名	Q/FX-KP-KI-J
形名コード	13JC17
SH(名)-080735-R(1306)KWIX	

この印刷物は2013年6月の発行です。なお, お断りなしに仕様を変更することがありますのでご了承ください。 2013年6月作成

この標準価格には消費税は含まれておりません。ご購入の際には消費税が付加されますのでご了承おき願います。 標準価格 3,000円